UNIVERSITY OF OSLO
Department of Physics

Design of a Prototype Communication System for the CubeSTAR Nano-satellite

Johan L. Tresvig

July 2010

Abstract

This thesis describes the design and implementation of a prototype communication sub-system for the CubeSTAR satellite.
The report presents the realization of a semi-duplex UHF transceiver compatible to the GENSO network. Applicable antenna solutions for the satellite is discussed with regards to directivity, polarization, operational redundancy and payload requirements, a double dipole antenna configuration is proposed for the satellite.
The CubeSTAR satellite is a student satellite project initiated at the University of Oslo in late 2008. The satellite will carry a scientific payload called *multiple Needle Langmuir Probe*. The m-NLP is an experimental instrument designed at the UiO. The instrument is used to measure the electron density in the ionosphere. The m-NLP is previously been tested at ESTECs plasma lab and flown on the ICI-2 sounding rocket.

Acknowledgement

This thesis has been written at the Department of Physics at the University of Oslo during the period from September 2009 to July 2010 under the supervision of Associate professor Torfinn Lindem.

I would like to thank Professor Torfinn Lindem for his full support and guidance through this period, for always motivating and pushing me forward and for allowing me to participate in this exciting project. I am also grateful for all the help and guidance provided to me by the electronic workshop in building the system and particularly Stein Lyngen for his patience, enthusiasm and vast knowledge of PCB and RF design.
I am deeply thankful for the great team work with the other CubeSTAR team members and particularly the members of the communication team, Henning Vangli and Markus Grønstad whom I have shared with countless hours of fun and frustrations.
I am deeply appreciative to the European Space Agency and the Norwegian Space Center for providing me with the financial support to attend the Space Studies Program in California during the summer 2009.
I would also like to thank my friend Thang Le Nguyen and my uncle Erling Schøller for helping me with spell checking and making the language in the report understandable.

Last but not least I would like to thank my family, my parents and brother for their continuous support during this thesis and in all my studies.

Norway, Oslo, July 2010
Johan L. Tresvig

Contents

Chapter 1

Introduction

1.1 CubeSTAR

The CubeSTAR project is a student satellite project at University of Oslo (UiO) in Norway. The project was initiated in December 2008 by the Department of Physics at UiO and the Norwegian Centre for Space-related Education (NAROM) with financial support from the Norwegian Space Center (NSC).
The CubeSTAR is a nano-satellite which is being built after the *Cubesat* standard. Cubesat is a satellite standard developed by California Polytechnic University (Calpoly) and Standford University in 1999. The standard specifies a mechanical structure with the physical dimensions, 10x10x10cm, with a maximum weight up to 1.33kg. The unit is called "1U". This standardization has made launches relatively cheap and consequently it has become a popular standard for university satellite projects. The CubeSTAR will be built as a "2U", which means that the satellites physical dimensions will be 10x10x20cm and its weight will be no more than 2.66 kg (see figure 1.1).
The project is divided into six work groups, one for each of the satellite subsystems and a project management team:

- Electronic Power System
- Communication
- Payload
- Attitude Determination and Control System
- On-Board Data Handling
- Project Management

The groups are managed by students, where the work is mainly performed through master thesis.
The students are responsible for developing the various systems and solving all the technical aspects of the project, but will receive support from the staff from the faculty (project leader, technical and scientific advisors), the electronic workshop and the mechanical workshop.

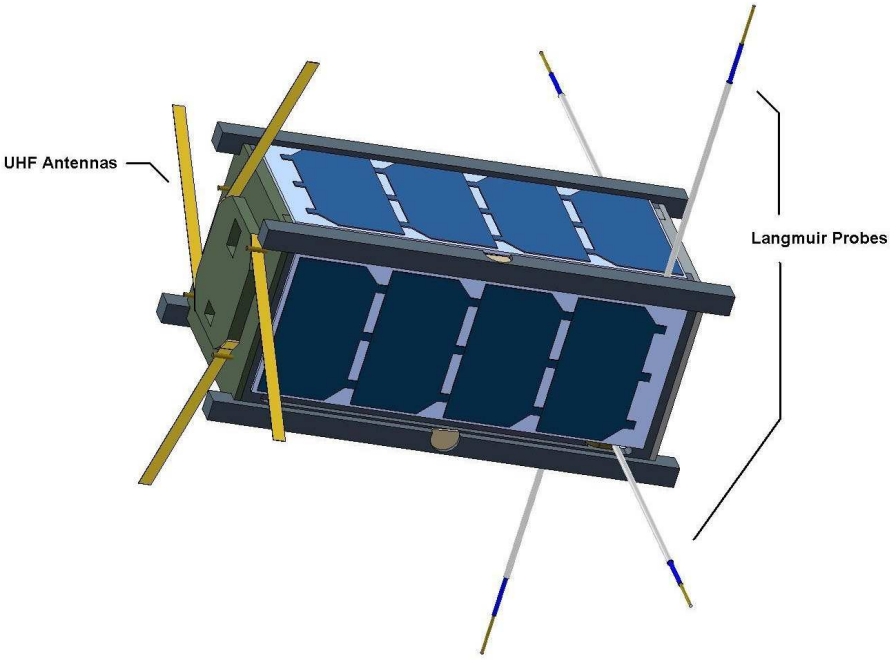

Figure 1.1: Figure of the CubeSTAR structure

1.1.1 ANSAT

The Norwegian Student Satellite Program (ANSAT) was initiated by the NAROM, NSC and Andoya Rocket Range (ARR). The aim of ANSAT is to create a student satellite program [1] and to launch three to four student satellites within a five year time frame. The first satellite, built by the Technical College of Narvik, is called HiNCube and is scheduled to be launched in late 2010. The CubeSTAR is the second satellite in the program.

1.1.2 Scientific Experiment

The CubeSTAR will carry a scientific experiment as payload. The experiment is called multiple Neddle Langmuir Probe (m-NLP) which is an instrument designed to measure electron density in the ionospheric plasma [2].

The probes are based on a new concept that will increase the spatial resolution to a few meters. Measuring the electron density is of interest for space weather monitoring over the polar cusps to improve communication and navigation in those regions. The instrument has previously been tested at ESTECs Plasma Lab and flown on the ICI-2 sounding rocket. Flying the instrument on CubeSTAR is the next step to prove the technology and gain flight heritage for the instrument.

1.2 Goals of the Thesis

The purpose of this thesis is to define and develop a prototype of the communication system in the CubeSTAR satellite. The prototype system includes the hardware (Printed Circuit Board (PCB)) and the software drivers for the system. The software drivers are required to abstract the hardware issues in the system such that the protocol layer [3] may interface with the system as a generic communication channel. The thesis will also include a survey and an evaluation of possible antenna solutions applicable for the CubeSTAR satellite. The goal of the thesis is summarized into four points:

- Define the requirements of a communication system for the CubeSTAR satellite
- Design and implement a prototype system which meets these requirements
- Develop necessary software drivers to interface the prototype system to the protocol layer
- Discuss and propose an antenna solution for the CubeSTAR satellite

1.3 Report Outline

This report is a master thesis as well as technical documentation for the CubeSTAR communication system (space segment). The report will be outlined to provide future members in the CubeSTAR project a thorough understanding of the work that has been done. For the same reason some of the references in this document are not directly linked to the work in this thesis, but are meant to suggest important background reading for future CubeSTAR team members.
The report will first give an introduction of CubeSTAR project in chapter 1. Chapter 2 and 3 identify and perform an analysis of the requirements for the communication system. In chapter 4 the system architecture and the practical implementation of the system is presented.
Chapter 5 presents the proposed antenna solutions that have been identified

and the correspondent options are discussed. In chapter 6 the testing and validation of the prototype system is presented.
Chapter 7 will give a discussion of the problems encountered, future works and recommend improvements for the next version of the system. Chapter 8 will give a summary of the work, the findings and general conclusion.
Appendix A, B and C contains the relevant theory to understand the terms, principles and methodology discussed in this paper.

Chapter 2

System Requirements

This chapter will identify the requirements for the CubeSTAR communication system. The chapter will introduce some fundamental satellite communication terms, which are explained in greater detail in appendix A and in footnotes.

2.1 Functionality

A Cubesat communication system has three primary functions, (1) to transmit a tracking signal, (2) download telemetry to a ground station and (3) to receive commands from a ground station. A satellite communication system is often referred to as a TT&C (Tracking, Telemetry and Command) system after these functions.

Beacon

A tracking signal, also known as a *beacon*, makes it possible for a ground station operator to find the position of a satellite moving over the sky. The signal will help the operator to locate and maintain optimal antenna pointing towards the satellite during each pass. The signal is automated and transmitted periodically from the satellite to the ground station.
The beacon serves a secondary task as well: It integrates vital information on the satellites status and thus furnishes the satellite team with information from the satellite even if the team is unable to communicate with the satellite.
A beacon will typically transmit the satellites name along with some vital data on the satellites status (e.g. battery conditions, internal temperatures, etc).

Data Link

A data link is a two way communication channel, where the *uplink* is the communication from the ground station to the satellite and the *downlink* is the communication from the satellite to the ground station.
The downlink is used to transmit telemetry, typically measurement data from the Langmuir probes and housekeeping[1] data and so forth.
The uplink is used to send commands to the satellite.

2.2 Communication Protocol

A satellite and a ground station must use the same communication protocol to be able to communicate with each other. A communication protocol implies a standardization of frequency, bandwidth, data rate, modulation scheme and packet protocol.

The CubeSTAR satellite will orbit in a Low Earth Orbit (LEO) which makes its *footprint*[2] relative small, and limits the time the satellite can communicate with the ground station in each orbit. An effective way to mitigate this constraint is to increase the number of ground stations a Cubesat team can use to communicate with their satellite, the best way to do this is to become member of a Ground Station Network (GSN). The Global Educational Network for Satellite Operation (GENSO) is a GSN made particularly for the Cubesat community. It was decided early in the design phase that the CubeSTAR satellite should be compatible with the GENSO.

2.2.1 GENSO

The GENSO network allows its users to operate other Cubesat ground station through the internet increasing the time a Cubesat team can communicate with their satellite. Since the GENSO network is a software standard[4] it does not impose any requirements on the communication system directly, however GENSO has created a reference ground station[5] which also the CubeSTAR ground station is designed after. In the following discussion of the parameters for GENSO compability the ground station will be used as reference.

2.2.2 Packet Protocol

The GENSO project has proposed the AX25 as packet protocol
The AX25 is designed for radio amateur usage and is often used in the am-

[1] Housekeeping data is internal information about the status and health of the satellite or one of its subsystems

[2] The footprint is the area the satellite has communication coverage on Earth at any given moment

ateur radio packet networks.
A description of the protocol can be found here [6]. The communication protocol implemented in the CubeSTAR communication system is a simplified version of the AX25 packet protocol, the protocol is described in detail in [3].

2.2.3 Frequency Band

The GENSO ground station uses the ICOM910H radio. The radio is intended for use within the Very High Frequency (VHF), Ultra High Freqency (UHF) and S-band.
The CubeSTAR ground station has not been built for S-band communication, because of this the option for communication on the S-band was omitted. VHF and UHF is defined by ITU as the radio frequency band between 30MHz-300MHz and 300MHz-3GHz. Within the CubeSat community it is common to use the amateur satelitte radio frequencies as they do not require a formal application to the International Telecommunication Union (ITU).
The usage of VHF and UHF frequencies for amateur satellite operations is regulated by the Norwegian Post and Telecommunication Authority (NPT) and can be found in the Norwegian frequency allocation map [7]. The NPT has also defined the maximum available bandwidth in the regulation for radio amateur license [8].
A summary of the regulation imposed by the NPT for amateur satellite operation:

- VHF, 144-146MHz, $B_{max} = 18kHz$
- UHF, 434.79-438MHz, $B_{max} = 30kHz$

Due to the constraints imposed by the payload, (see 2.5 for further discussion) and the larger bandwidth available in the UHF band the communication system should operate in the 434.79-438MHz frequency band.

2.2.4 GENSO Recommended Radio Configuration

The GENSO project has recommended two radio configurations [9]:

- 1200bps using Audio Shift Keying (AFSK) modulation (see subsection A.3.2).
- 9600bps using Frequency Shift Keying (FSK) modulation (see subsection A.3.1).

The baud rate is constrained by the available bandwidth of the system. The maximum bandwidth the system can occupy is 30 kHz (see 2.2.3).
International regulation (see subsection 2.3.1) determines that the assigned bandwidth should include both the data signal B_{sig} and two times the

Doppler shift Δf introduced into the signal due to the speed of the satellite relative to the Earth.
Using Eq. 2.3 and the figure for expected Doppler shift calculated in 2.3.2 it is possible to determine the limit of the signal bandwidth.

$$\begin{aligned} B_{occ} &= 2\Delta f + B_{sig} \\ B_{sig} &\leq B_{max} - 2\Delta f \\ B_{sig} &\leq 30kHz - 2 * 10.5kHz \\ B_{sig} &\leq 9kHz \end{aligned}$$

From the calculation above it is determined that the bandwidth of the signal most be $\leq 9kHz$.

Calculating the Bandwidth of a 9600 baud Signal

The bandwidth of a FSK modulated signal is given by Carson's rule (see Eq. A.15).
Following the recommendations from the International Telecommunication Union - Radiocommunication Sector (ITU-R)[3] on bandwidth calculations [10] the Carson's rule has added a constant (k).

$$B_{fsk} = 2(k\Delta f + \frac{baud}{2}) \tag{2.1}$$

where
k =1.2

Calculating the B_{sig} for a 9600 baud signal, where Δf is set to 3 kHz (see [3]):

$$\begin{aligned} B_{sig} &= 2(1.2 * 3kHz + \frac{9600baud}{2}) \\ B_{sig} &= 16.8kHz \end{aligned}$$

The calculations shows that bandwidth of a 9600 baud signal is larger than the maximum bandwidth available. Thus this radio configuration can not be used.

Calculating the Bandwidth of a 1200 baud Signal

Calculating the B_{sig} for a 1200 baud signal, where Δf is set to 0.5 kHz (see subsubsection below):

$$B_{sig} = 2(0.5kHz * 0.6kHz + \frac{1200baud}{2})$$

[3]THE ITU-R, is a division of the ITU. The purpose of the ITU-R is to manage and harmonize the international radio- frequency spectrum and satellite orbits to ensure that radio systems do not interfere with each other

$$B_{sig} = 2.2kHz$$

The calculations shows that bandwidth of a 1200 baud signal is within the maximum bandwidth limit.

Implementing AFSK onto an FSK Transceiver

AFSK is a hybrid modulation scheme using analog signals to emulate digital high and low level. The modulation scheme uses two tones, 1200Hz and 2200Hz to signal low and high. "True" AFSK can not be implemented on a FSK transceiver since it operates with digital signals. However using an algorithm described in [27] a FSK modulated signal can be decoded as a AFSK at the ground station.
The frequency difference between the two tones used in AFSK modulation is 1kHz, by transmitting a FSK modulated signal with a $\pm 500Hz$ frequency separation the ground station can receive the signal in Lower Side Band (LSB) mode and extract two tones close to 1200Hz and 2200Hz.
The preselected transceiver, the CC1101 (see subsection 4.1.2) do not support a frequency deviation lower than 1.6kHz. Due to this fact the radio configuration using 1200 baud signal AFSK can not be used in this design.

2.2.5 Custom Radio Configuration

The previous subsection discussed and showed that the two recommended radio configuration was not applicable due to bandwidth limitations and technical constraints imposed by the transceiver chip. The GENSO network does not impose hardware requirements, and such allows for custom radio configurations to be used in the network. Because the two recommended radio configurations proposed by GENSO could not be supported by the system, it was necessary to define a custom radio configuration for the communication system.
The FSK modulation scheme was selected, using a Gaussian filter to reduce the spectral width of the radio signal. The maximum signal rate was calculated:

$$\begin{aligned} B_{sig} &= 2(k\Delta f + \frac{signal\,rate}{2}) \\ 9kHz &= 2(1.2 * 1.6kHz + \frac{signal\,rate}{2}) \\ signal\,rate &= 9kHz - 2(1.2 * 1.6kHz) \\ signal\,rate &= 5160\,baud \end{aligned}$$

The signal rate was approximated to the nearest standard signal rate, 4800 baud.

The radio configuration for this system is defined:

- 4800 baud
- Gaussian Frequency Shift Keying (GFSK) modulation scheme

2.2.6 Beacon

The beacon signal is an independent radio signal not constrained by GENSO. In the Cubesat community it is common to use a beacon signal which transmits a Morse coded signal (see subsection A.3.3) using a Continuous Wave (CW) modulation scheme.
A beacon will typically transmit on low data rates (approx. 10-15WPM) so the signal can easily be picked up by radio equipment. This technique is popular because it requires less RF signal power and can be received by simple radio equipment and does not require decoding equipment to decode the data.

2.3 Radio Regulations

Because the communication system emits radio energy, it is important that the system follows the governing standards for radio transmissions in order that it does not interfere with other transmissions on nearby frequencies.
The purpose of radio communication regulations is to coordinate the usage of various frequencies in order to prevent the transmissions interfering with each other. The overall governing regulations for this kind of communication system is the ITU-RR Vol.1-4 (the part pertaining amateur radio operation can be found in [11]). This document defines general terms and recommendations for regional and national regulatory bodies.
Two major factors which regards the harmonization of radio regulations, (1) keeping the wanted signal within the allocated bandwidth and (2) reducing the unwanted emissions outside the bandwidth as much as possible.

2.3.1 Maximum Bandwidth

The NPT has defined the maximum bandwidth for amateur radio operations in the UHF band to be 30 kHz. The assigned bandwidth shall include the bandwidth of the data signal (B_{sig}, given by equation 2.3) and two times the Doppler shift (Δf, given by equation A.14) introduced by the velocity of the satellite (see Radio Regulations, Vol.1, Art.1, Sec.VI, pt. 1.147-1.152).

$$B_{sig} = \frac{Baud}{2} \tag{2.2}$$

The total bandwidth (B_{occ}) is given by:

$$B_{occ} = B_{sig} + 2\Delta f \tag{2.3}$$

2.3.2 Doppler Shift

Doppler shift is a phenomenon occurring when the transmitter is moving relative to the receiver. Since the satellite is orbiting the Earth moving at great speed this can cause cause the satellite to transmit outside the allocated bandwidth if not accounted for.
The Doppler shift is calculated in five steps:

- 1 Find the period one orbit
- 2 Find the speed of the satellite
- 3 Find the speed component between the satellite and ground station
- 4 Find the relative speed between the satellite and the ground station
- 5 Find the Doppler shift

The calculation below is made assuming a circular orbit, a satellite height of 400km above the Earth's surface and carrier frequency of 437MHz.

Orbit time

The orbital time is found using Kepler's third law (see subsection A.1.1):

$$\begin{aligned} T_{orbit} &= \sqrt{\frac{4\pi^2 a^3}{\mu}} \\ T_{orbit} &= \sqrt{\frac{4\pi^2 (6378km + 400km)^3}{3.986004418 * 10^5}} \\ T_{orbit} &= 5553.5s = 92.56min \end{aligned}$$

where
$a = R_e + height$
$R_e = 6378km$, the radius of the Earth

Satellite Speed

In order to find the speed of the satellite it is necessary to first find the circumference of the orbit d_{orbit} and divide the distance on the orbital time T_{orbit}:

$$\begin{aligned} d_{orbit} &= 2\pi a \\ d_{orbit} &= 2\pi (6378km + 400km) \\ d_{orbit} &= 42587.4km \end{aligned}$$

The speed v_s:

$$\begin{aligned} v_s &= \frac{d_{orbit}}{T_{orbit}} \\ v_s &= \frac{42587.4km}{5553.5s} \\ v_s &= 7668.6m/s \end{aligned}$$

Relative Speed

The relative speed v_r is the speed of the satellite relative to the ground station.
The speed v_r:

$$\begin{aligned} v_r &= v_s cos\theta = v_s * \frac{R_e}{R_e + h} \\ v_r &= 7668.6m/s * \frac{6378km}{6378km + 400km} \\ v_r &= 7216m/s \end{aligned}$$

Doppler Shift

The Doppler shift Δf is found using eq.A.14:

$$\begin{aligned} \Delta f &= f_t \frac{v}{c} \\ \Delta f &= 437MHz \frac{7216m/s}{3 * 10^8 m/s} \\ \Delta f &= 10511Hz \end{aligned}$$

The calculations shows that any ground station situated at the edge of the satellites communication coverage will experience a Doppler shift close to $\pm 10.5kHz$, assuming that the satellite orbits the Earth in circular orbit 400 km over the ground and transmits on 437MHz.

2.3.3 Unwanted Emissions

Unwanted emissions consist of two types of emissions, *Out-of-Band (OOB) emissions* and *spurious emissions*.
OOB emissions are emissions that occur just outside the assigned bandwidth due to the modulation process. Spurious emissions consist of harmonic emissions [4] and parasitic frequencies [5]

[4] Harmonic emissions, are frequency components which is a multiple of the transmitted frequency

[5] Parasitic frequencies, are randomly generated frequency components outside the OOB domain

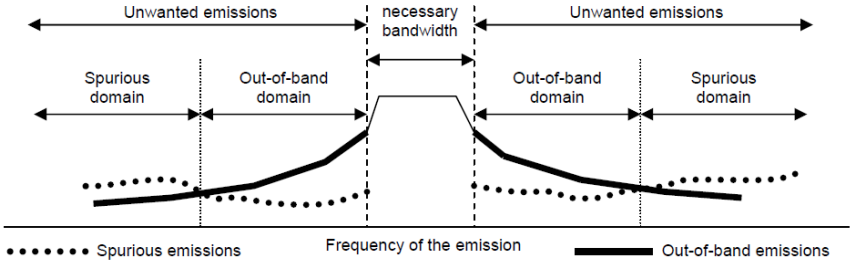

Figure 2.1: Unwanted emissions and necessary bandwidth

2.3.4 Termination of Transmissions

According to ITU-R regulations the system shall also be able to cease any radio transmissions through commands from the ground station (see Radio Regulations, Vol.1, Art.22, Sec.I, pt. 22.1, §1).
This functionality must be implemented in the command protocol for the satellite communication system, but lies outside the scope for this thesis and will not be addressed further.

2.4 Mechanical Requirements

A Cubesat structure is a clearly defined mechanical structure. It is important that the system (e.g. PCB) fits into the mechanical structure of the satellite. Thus the system must meet the physical and mechanical requirements for the PCB and bus connector given by the CubeSTAR structure team. The mechanical requirements of the system was provided by the University of Oslo's Electronic Workshop (ELAB) which is responsible for the mechanical fitting of the electronics in the satellite. The mechanical requirements [12]:

- 80 x 75 mm (w x h)
- Build height $<$ 25 mm

2.5 Payload Requirement

The scientific experiment is designed to measure the electron density in the near surroundings of the satellite. It is therefore highly sensitive to electromagnetic interference. The communication system can interfere in two ways, through transmissions and by creating wake turbulence.

2.5.1 Transmissions

During transmissions the energy output from the antennas will disrupt all scientific measurements. This can not be avoided, but the system should incorporate some type of warning to the On Board Data Handler (OBDH) while transmitting in order that the m-NLP system cancels all measurements while the transmitter is active.

2.5.2 Turbulence

Whenever the satellite is flying through the ionospheric plasma (see 1.1.2) the satellite structure can cause turbulence in the plasma. This effect is mitigated by situating the probes on deployable booms extending out from the satellite structure.
However if the satellite is oriented in such a way that the antennas lies in front of the Langmuir probes in the forward direction, the antennas can create turbulence which will interfere with the scientific measurements. Because of this it is important to minimize the length of the antennas. The length of the antennas is roughly the same length as the wavelenght of the signal (see equation C.1). By increasing the frequency of the signal the length of the antennas is reduced.

2.6 Environmental Requirement

Space offers a harsh environment for electronic circuits. Satellite systems are therefore subjected to many design challenges. This section will discuss the key factors to be considered in the design.

2.6.1 Radiation

In space, an electronic system is exposed to many radiation types like high energy ion radiation, magnetic fields and plasma interactions. The effects of radiation can be diverse ranging from non-destructive memory corruption known as "bit-flips", degradation of function to permanent damage of components and systems.
The most common events are Single Event Upset (SEU), Single Event Latchup (SEL) and Total Ionization Dose (TID) which can influence electronic devices in various ways.

Single Event Upset

SEU, is an event where a charged particle hits a logic gate and alters the value. This is referred to as an *soft error* because it causes a bit-flip, but does not permanently damage a device.
SEU is the most likely event to occur. SEU is mainly a software problem

and can be mitigated using memory scrubbing.
Memory scrubbing is a process in which the program memory is checked for SEU using CRC checks. If a SEU is detected the program memory will be overwritten by a "healthy" copy of the program code. This requires however several copies of the program code aboard the satellite occupying memory resources and Tripple Modular Redundancy (TMR) which is a method where a calculation is run on three parallel processes. The three results are compared and the results which gave most identical answers are passed along. Since SEU is a software issue it will not be covered in this thesis, however future software development should consider this effect.

Single Event Latchup

SEL is an event where a charged particle passes through a "parasitic" thyristor (a common circuit in CMOS design) and causes a short circuit in the device, permanently damaging the device, this is known as a *hard error*.
SEL is an effect which can cause damage to the hardware in the system. It is common to mitigate the effects SEL by introducing current limiting circuitry in the system. It is practice to also add stand-by redundancy.
Stand-by redundancy is a design strategy to insure that a system can continue to operate even if a system fails by including a back-up system to take over if the primary system should fail. E.g. the communication system could have two transceiver systems, one powered down while the primary is functioning. If the primary transceiver fails, the back-up transceiver can resume communication.
The thesis will only cover a prototype system with focus on functionality so any back-up redundancy should be considered in future development of the communication system. Nevertheless current-limiting circuitry should be added to this design.

Total Ionization Dose

TID is an effect which takes place over time as a device accumulates radiation the performance of the device degrades. TID effects is primarily mitigated using Rad-hard components and shielding.
Rad-hard are components designed particularly to withstand radiation. They are primarily used in military and space applications. Due to the limited operational time of CubeSTAR (3-6months) this effect will not be considered a constraint for this project.

2.6.2 Vacuum

The satellite is expected to be released in an orbit 300 to 600km above the ground. In such altitudes the atmospheric pressure can be considered a

vacuum. In a vacuum environment, electronic circuits can experience mechanical deformation and out gassing. The components used in the communication system must be selected with this in mind.
The process of mounting components onto the PCB may introduce air bobbles into the solder. This can cause the solder to detach it self from the attachment pad in when the surronding air pressure is reduced.

2.6.3 Temperature

The satellite will experience a considerable temperature range during launch and in orbit. The electronic components has to be chosen by their operating temperature to meet the temperature requirements.
A thermal analysis of the CubeSTAR satellite has not been performed, but conclusions made by other Cubesat teams [13],[14] suggest that the components should be able to operate from -40^o and -30^o up to between $+40^o$ and $+85^o$. The values will be used as guidelines for component selection.

Chapter 3

Link Budget

This chapter will describe the link budget analysis, discuss the key parameters used and the results provided by the calculations. The calculations and the summary can be found in appendix E.

3.1 Introduction

A link budget is an analysis tool used to determine if a communication link is applicable given key parameters such as transmitted signal power, frequency, data rate and the bandwidth in the communication link.
The link budget calculates the gain and loss of a RF signal from the modulation in the transmitter to the demodulation in the receiver. The result is a value known as Signal to Noise Ratio (SNR).

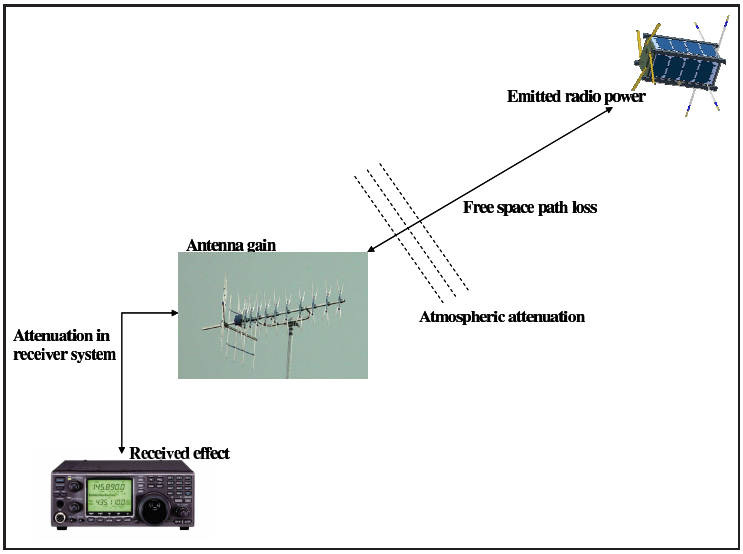

Figure 3.1: A satellite link

3.1.1 E_b/N_0

E_b/N_0 is the ratio between energy per bit and noise per hertz. The first step in a link budget analysis is to determine the required E_b/N_0 at the receiver input. The required E_b/N_0 can be identified using graphs found in communication text books where f(BER, modulation scheme) = E_b/N_0.
The E_b/N_0 is a normalized value of the SNR, it is common to use this value to compare digital communication links.

3.1.2 SNR

SNR is the ratio between signal power and noise power (P_s/P_n) at the receiver input. This measure is preferred when calculating link budgets for wireless communication systems to the E_b/N_0 value because the system uses power waves with inifinite duration.
SNR is determined by two calculations, the received signal effect (P_s) (see Eq. A.7) and the received noise effect P_n (see Eq. A.11).
From the SNR one can determine the E_b/N_0 in the actual system using equation 3.1 and thus determine if the required E_b/N_0 is met and subsequently if the communication link can meet the specified Bit Error Rate (BER) value.

$$\frac{S}{N} = \frac{P_s}{P_n} = \frac{E_b}{N_o}\frac{f_b}{B} \tag{3.1}$$

where
f_b = bit rate
B = bandwidth
If the calculated E_b/N_0 is equal to or greater than the required E_b/N_0 it is said that the link closes.

3.1.3 Bit Error Rate

BER is a measure of the quality of a communication link. The value indicates the statistical probability for a bit transmitted through the communication link to be received with the wrong value. E.g. a BER value = 10^{-6} indicates that only one in a million bits transferred through the communication link will be received with wrong value. The BER value is dependent on the E_b/N_0 value at the input of the receiver.

3.1.4 Link Margin

A link margin is an error margin put into the link budget to mitigate unforeseen attenuations like cloud cover, antenna pointing errors, rain, unexpected noise sources, etc.
No stated value has been found, but recommendations from IARU/AMSAT

and local radio amateurs suggests that the link margin should be approximately 10-12dB on top of the required SNR value in order to be certain that the communication link closes.

3.2 Link Budget Parameters

This section will discuss and define some of the key link budget parameters of the system.

3.2.1 Effective Isotropic Radiated Power

Effective Isotropic Radiated Power (EIRP) is the signal power level emitted from the transceiver. The EIRP is the sum of the effect supplied to the antenna and the gain in the antenna.

Transmitted Power

A transmission from the satellite to the earth can be one of the most current consuming processes in a Cubesat satellite. Due to the early design phase pf the satellite, it has not been possible to estimate an accurate current budget for the satellite, but it has shown that it was necessary to look at other Cubesat projects.
A paper published in late 2008 [15] lists different communication systems used on various Cubesat satellites used until 2008 and seems to give reliable indications of the amount of power the TT&C system should transmit. The paper shows that except for one commercial Cubesat all Cubesats transmitted 1000mW or less. Taking into account some signal attenuation between the output of the High Power Amplifier (HPA) and the antenna, the power from the HPA is set to 1000mW or 30dBm in the link budget.

Antenna Gain

The CubeSTAR satellite is assumed to use a near isotropic antenna (see section 5.2) to account for tumbling of the satellite. The gain of a dipole antenna is approximately 2dB.

3.2.2 Path Loss

The total loss of signal power is called the path loss. The path loss is determined by the free space path loss and the atmospheric attenuation.

Free Space Path Loss

The free space path loss is caused by the reduction of flux density due to the distance the signal must travel (see subsection A.2.2) and the receiving

antennas ability to absorb the emitted energy. The free space path loss is determined by the maximum distance and the frequency of the radio signal.

The maximum distance (d_{max}) or *Slant range* between the satellite and the ground station is determined by the height of the satellite above the Earths surface and the minimum elevation angle on the ground station antenna. The distance can be calculated using equ. A.10.
The exact height of the orbit is not determined yet because the date of the launch has not been scheduled. In order to have a successful scientific experiment[2] with the payload a height between 600km and 300km is required. The height in the link budget was selected to be 600km to account for worst case condition. The eccentricity of the orbit is assumed to be close to zero(see section A.1.2).

The minimum elevation angle has a practical limitation between $5^o - 10^o$[16] due to terrestrial obstructions and thermal noise.
The CubeSTAR ground station is located on the roof of the faculty building. Due to several high power transmitters only a few hundred meters away belonging to the *The Norwegian Broadcasting Corporation* it was important to make sure the ground station antenna did not pick up interference from those transmitters. Although the antennas has a beamwidth (see subsection C.3.1) $= \pm 14^o$ which means that the minimum elevation angle has to be 14^o or above to avoid picking up radio energy from the nearby transmitters. In this link budget calculation the minimum elevation angle is set to 10^o as it is assumed that the small fraction of the beamwidth still able to pick up noise is negligible.

The frequency band selected for this communication system is 434.79-438MHz (see subsection 2.2.3).

Atmospheric Loss

Atmospheric losses are a generic term which includes several phenomena that can cause losses to a radio signal. Among them are polarization mismatch loss, rain attenuation [1] and refraction [2]. All of this phenomenas should be considered when making a communication link. However for satellite communication using frequencies from UHF and above the atmospheric losses are much smaller than the Free-Space Path Loss (FSPL) and will be absorbed by the link margin, see subsection 3.1.4.

[1] Rain Attenuation is a form of absorption, caused by rain drops, ice or snow

[2] Refraction is a phenomenon in which a wave changes direction when traveling from one medium into another.

3.2.3 Noise

The noise power introduced into the communication link is caused by thermal noise. Thermal noise is calculated using eq.A.11. In addition to calculating the noise power P_n at the receiver input the noise figure of the components between the antenna and the receiver must be added to find the actual S/N ratio at the receiver input.
In subsection several sources of electric noise sources are presented, most of this can be omitted due to the EMC shielding of the transceiver circuit (see subsection 4.3.4). Noise sources electrically connected to the CubeSTAR backplane bus might influence the transceiver circuit (like e.g. a switched power supply) and may reduce the S/N ratio. The effect of these kind of noise sources should be analyzed at a future date.

3.2.4 Receiver Gain

The transceiver system has an amplifier with a low noise figure between the antenna and the transceiver chip to increase the power of the received signal. The amplifier has a typical gain of 30dB.

3.3 Link Budget Calculations

The *AMSAT / IARU Annotated Link Model System* an excel spreadsheet was used to calculate the link budget The CubeSTAR communication team opted for a data link with a BER value equal to 10^{-5}. The link budget calculations can be found in appendix E.

3.3.1 Summary

The calculations produced the following link margins.
For the uplink:

$$SNR \quad = 25.5dB$$

For the downlink:

$$SNR \quad = 12.3dB$$

The calculated link margin is above the required link margin (see section 3.1.4) for both the uplink and the downlink. Assuming that the system can meet the requirements specified in this chapter the link should close and be able to maintain a $BER \geq 10^{-}5$.

Chapter 4

System Design

This chapter will present the hardware and firmware solution for the communication system.
The design methods will be discussed and the various subsystems within the communication system will be showed and explained. The PCB design and the layers, particularly the RF-design are presented.

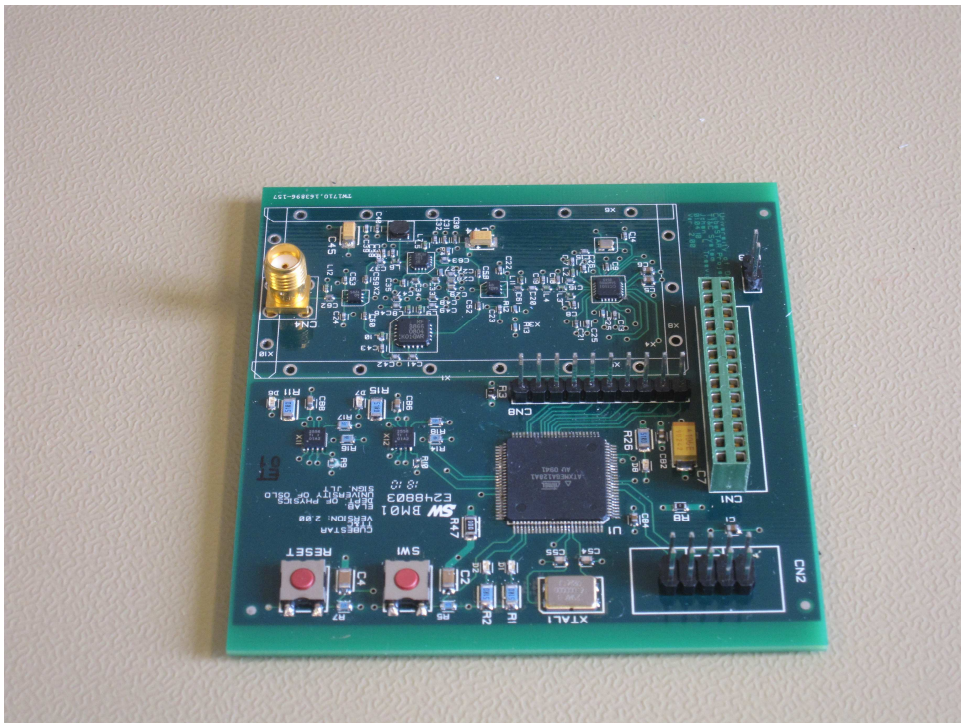

Figure 4.1: The PCB with components mounted

4.1 System Architecture

The system is built as a semi-duplex UHF radio transceiver. A semi-duplex transceiver is a two way system, but as opposed to a full duplex system, a semi-duplex system can not transmit and receive at the same time. As

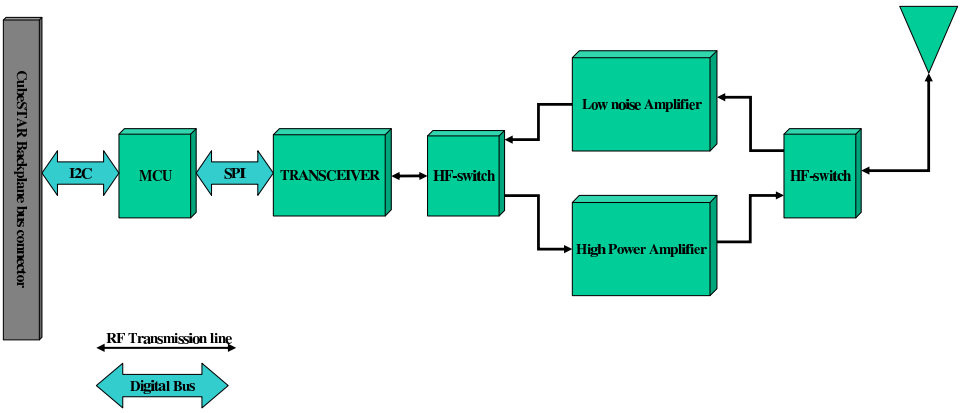

Figure 4.2: Block diagram of the communication system

shown in figure 4.2 the system is divided into several function blocks. This section will describe the function and system architecture in detail.

4.1.1 Micro Controller Unit

The MicroController Unit (MCU) (ATxmage128A1 from Atmel) is responsible for communication with the other subsystems in the satellite through the backplane bus. It decodes and encodes AX25 packets, controls the transceiver circuit and the radio front-end system. Figure 4.3 shows a block diagram of the MCU circuit.

I2C Bus

The I2C bus is used for internal communication with the other systems on the backplane bus. The bus is a 2-wire interface for low speed communication. This is the main communication bus in the satellite. The communication system will primarily use this bus to relay telemetry and commands to and from the OBDH. The bus runs on a clock rate of 400kHz.

UART Bus

The Universal Asynchronous Receiver/Transmitter (UART) bus is used as a debug interface for testing of the prototype system. The UART bus is connected to a UART/USB converter and the commands and feedback is displayed in a HyperTerminal window on a PC.

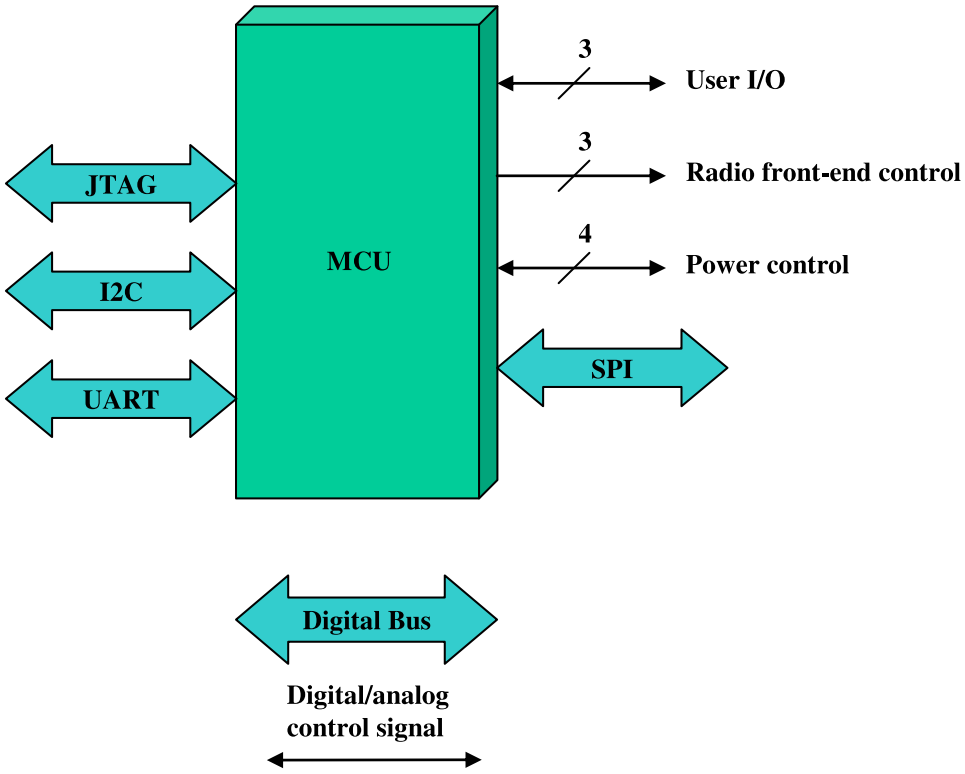

Figure 4.3: Block diagram of the MCU circuit

JTAG Interface

The Joint Test Action Group (JTAG) is included in the design to be able to debug and reprogram firmware on the MCU.

SPI Bus

The Serial Peripheral Interface (SPI) is used as interface between the MCU and the transceiver. The transceiver is configured through the interface and data exchange between the MCU and the transceiver is performed on this bus.

Control Signals

The MCU has dedicated signal lines used for control and sensing of various circuitry.

- Controls the position of HF-switches
- Controls gain in HPA
- Controls power to HPA and Low Noise Amplifier (LNA)

- Detects current consumption above the limit in HPA and LNA
- User interface, LEDs and switches

The signals are secured with pull-up/-down resistors and the system enters a default setting (safe mode) to protect the system in case the MCU looses control of the lines during reprogramming or due to a failure. When the system enters the safe-mode the HF-switches will be positioned in receive mode, power switches is turned off and the gain to the HPA is disabled.

4.1.2 Transceiver

The transceiver is responsible for reception and transmission of radio signals. The circuit is semi-independent, meaning that it can transmit and receive data independently, but the MCU must configure the transceiver before any operation and upload/download data from the internal data buffer when it is full or empty. The MCU communicates with the transceiver through the SPI interface. The transceiver is a vital part of the communication system.

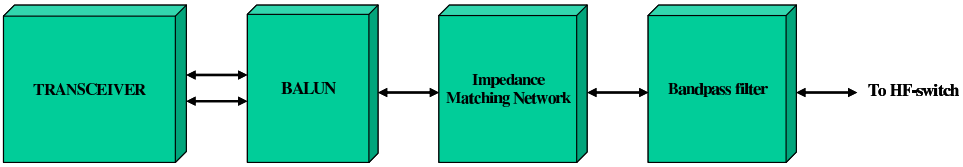

Figure 4.4: Block diagram of the transceiver circuit

the device converts data transferred from the MCU on the SPI interface into analog RF signals. The signals are modulated in accordance with the configuration uploaded from the transceiver. The transceiver can transmit both beacon and data signals. In receive mode the internal packet handler in the receiver will search for a *sync word* (a sequence of bytes) at the assigned center frequency (f_c). When the sync word is detected, in this case the AX25 start flag, the transceiver will start extracting data from the signal and alert the MCU that a valid signal has been received through an interrupt line.

CC1101

The transceiver system is built around the Texas Instrument CC1101 transceiver chip. The chip is a low-power RF transceiver operating in the required frequency band. The chip can de-/modulate a FSK and an On Off Keying (OOK)/ CW signal.
The transceiver can be operated in two modes. The first mode is through direct serial control receiving and transmitting a raw bit stream data through the GDx pins. This mode allows for a high degree of flexibility of packet handling since the MCU can be programmed to process the received data in

any number of ways. The drawbacks of this approach is that the MCU will use much of its resources to manage packet handling since this is a continuous operation.
The second mode is to use the built in functionality of the transceiver chip. This approach allows the system to use the built in packet handler to receive and transmit data. When the communication system is waiting for new data from the ground station the CC1101 can independently separate noise from data signals. The MCU is only alerted when the device has received a properly formatted signal. This approach will free the MCU to perform other tasks. Using the internal packet handler in the chip was selected for this design since the system is in receive mode approx. 90% of the time waiting for the ground station to initiate contact.

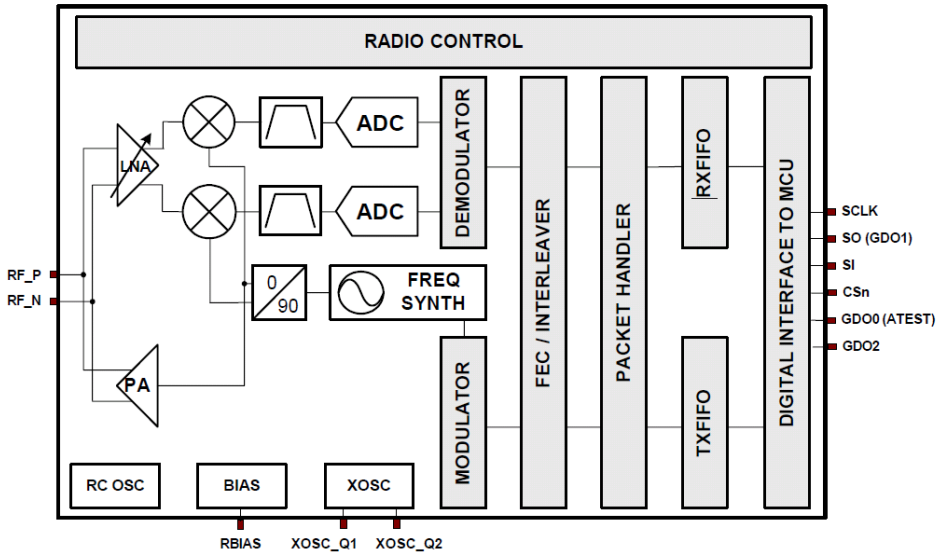

Figure 4.5: Block diagram of the transceiver chip

Balun

The balun is an electrical transformer converting an unbalanced signal into a balanced signal and vice versa. A balanced signal is a differential transmission line where a signal is referenced to another transmission line. This is often use to reduce the influence of external noise to a radio signal. An unbalanced signal, known as a single-ended signal is a radio signal referenced to ground.
The balun circuit acts as an transformer between the differential input on the transceiver chip and the single-ended transmission line in the radio-front end circuit.

Impedance Matching Network

The impedance matching network is designed to achieve characteristic impedance on the output of the transceiver (see subsection B.2).

Bandpass Filter

The bandpass filter rejects and suppresses frequency components outside the operational frequency band.

4.1.3 HF-Switch

The radio front-end system consists of two HF switches. The purpose of the switches is to switch the RF signal path between the LNA and the HPA. The input and output are matched internally to the characteristic impedance (see subsection B.2). To remove any DC component from the RF signal all the inputs and outputs on the switch has DC blocking capacitors.

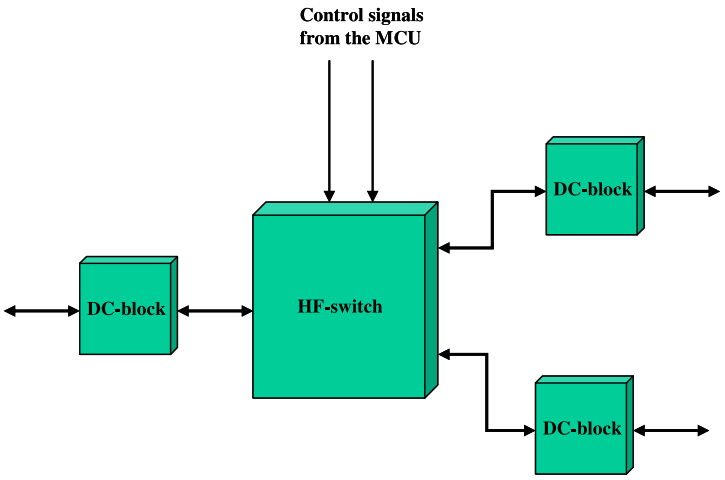

Figure 4.6: Block diagram of the HF-switch circuit

4.1.4 High Power Amplifier

The High Power Amplifier (HPA) is used to amplify the transmitted radio signal. The system is built around the RF5110G amplifier. The device is a 2-stage output amplifier, with a pre-amplifier and a driver stage. The amplifier has a 32.5dB gain and is capable of delivering up to 32dBm which equals 1500mW of output power (P_t).
The input is internally matched to the characteristic impedance however the output must have an impedance matching network.
The amplifier has adjustable gain which is controlled by the MCU.

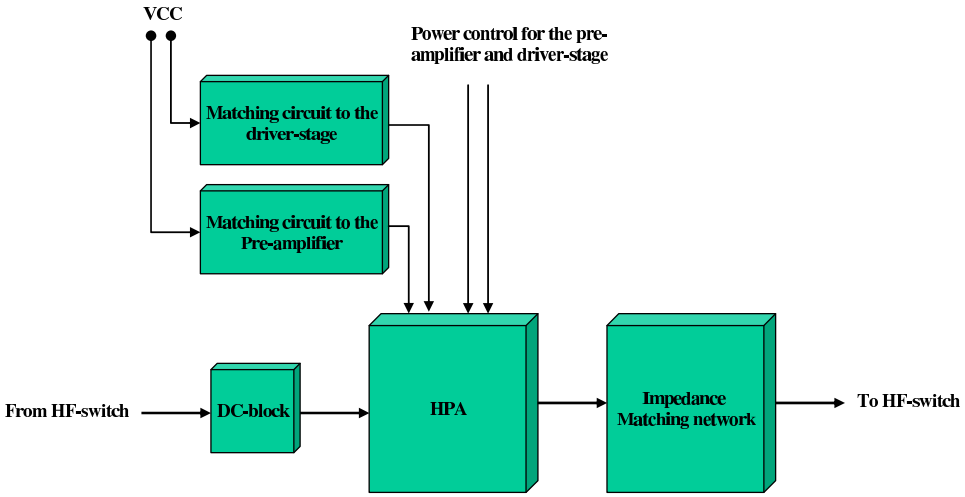

Figure 4.7: Block diagram of the High Power Amplifier (HPA) circuit

4.1.5 Low Noise Amplifier

The Low Noise Amplifier (LNA) is the input amplifier, which amplifies the received RF signal from the antenna into the receiver. The RF3866 low noise amplifier was selected due to the high gain (32.6dB) and low noise factor (1.75dB) (see subsection A.2.6). The input and output is matched to the characteristic impedance.

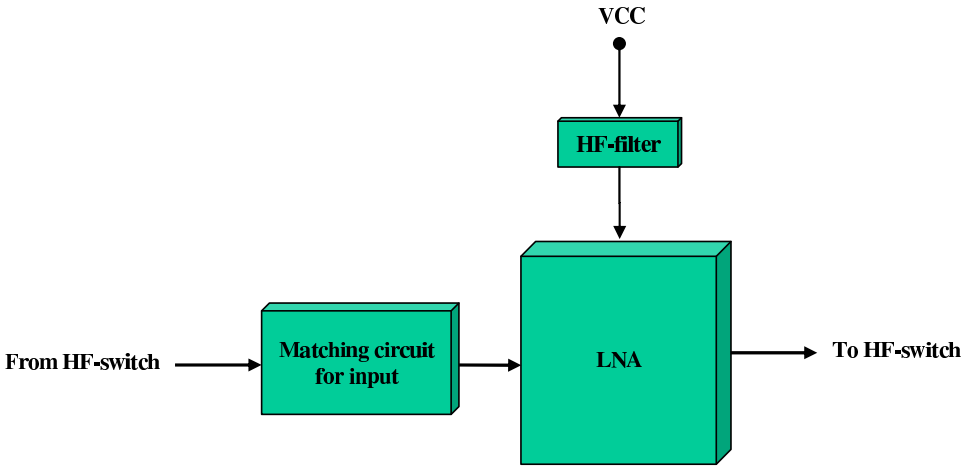

Figure 4.8: Block diagram of the Low Noise Amplifier (LNA) circuit

4.1.6 Power Switches

The power switches are used to control the power to the HPA and LNA. The switches, TPS2556 from Texas Instruments, functions as power distribution switches and over-current protection for the amplifiers. The switches are controlled from the MCU where each switch has an enable signal and an over-current indicator signal.
The over-current protection is triggered when the current passing through the switch exceeds the threshold set by the current limit (I_{os}) reference. If the limit is exceeded the switch will keep the output current at I_{os}.
if the switch detects a low voltage condition, $V_{cc} < 2.4V$ the switch will automatically switch off. The over-current protection in the circuit is designed

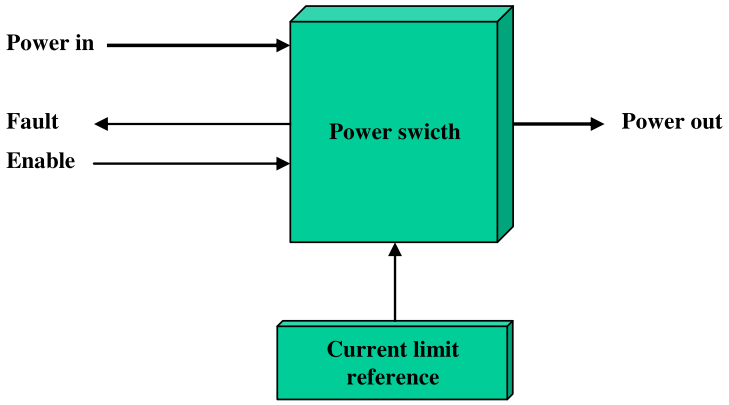

Figure 4.9: Block diagram of the power control circuit

to meet the maximum current of the HPA and the LNA. The formula in the datasheet for the TPS2556 is:

$$R_{ILIM(HPA)}[k\Omega] = \left(\frac{99038}{I_{max}[mA]}\right)^{\frac{1}{0.947}} \tag{4.1}$$

Using maximum current for $I_{Max(HPA)} = 1900mA$ and $I_{Max(LNA)} = 500mA$:

$$\begin{aligned} R_{ILIM(HPA)} &= \left(\frac{99038}{1900}\right)^{\frac{1}{0.947}} = 65k\Omega \\ R_{ILIM(HPA)} &= \left(\frac{99038}{500}\right)^{\frac{1}{0.947}} = 266k\Omega \end{aligned}$$

Both values where approximated to the closest resistor value:

$$\begin{aligned} R_{ILIM(HPA)} &= 65k\Omega \approx 68k\Omega \\ R_{ILIM(HPA)} &= 266k\Omega \approx 270k\Omega \end{aligned}$$

4.2 RF Design Methods

Designing a high frequency circuit requires a different design approach than standard electronic design. In standard electronic circuits it is common to use the *lumped element model*, where passive components has a constant value and transmission lines are considered "perfect" conductors. The lumped element model applies where the wavelength (λ) of the operating signal is much larger than the physical dimension (L_c) of the circuit, $L_C << \lambda$.
In a high frequency circuit the wavelength of the operating signal becomes much smaller, and the $L_C << \lambda$ is no longer true. Thus the lumped element model must be abandoned for the *distributed element model*. The distributed element model states that transmission lines has impedance and the value of passive components are dependent on the operating frequency.

4.2.1 2-Port network

The transceiver system is built using the 2-port network design approach (see section B.3) to uphold the *optimal power transfer* principle (see section B.1). Each of the active devices in the RF-design (transceiver, HF-switches, LNA and HPA) is viewed as a 2-port network and has an output and a input impedance equal to the characteristic impedance (see section B.2), $Z_{in} = Z_{out} = 50 + j0\Omega$. Figure 4.10 shows how the 2-port network approach is applied to the transceiver system. The two HF-switches is designed to switch the signal path between the LNA and the HPA depending if the system is receiving or transmitting. This creates two independent chains of 2-port networks.

4.2.2 Transmission Lines

In an 2-port network design the transmission lines (the cooper traces between each port) also has to be considered as 2-port networks (signal and ground) since the distributed element model states that transmission lines must be regarded as an impedance. However if the length (l) of the transmission lines are kept below 5% of the operational wavelength of the signal, it is safe to assume that the voltage level is the same across the entire length [17].
This means that the transmission lines can be viewed as lumped elements. In a *lumped element model* transmission lines can be considered "perfect" conductors. By using equation 4.2 and 4.3:

$$\lambda_{eff} = \frac{\lambda_0}{\sqrt{\varepsilon_r}} \tag{4.2}$$

$$l_{max} = \frac{5\%}{100\%} * \lambda_{eff} \tag{4.3}$$

where λ_0 is the wavelength in vacuum, λ_{eff} is the effective wavelength through a specific material other than vacuum and ε_r is the dielectric con-

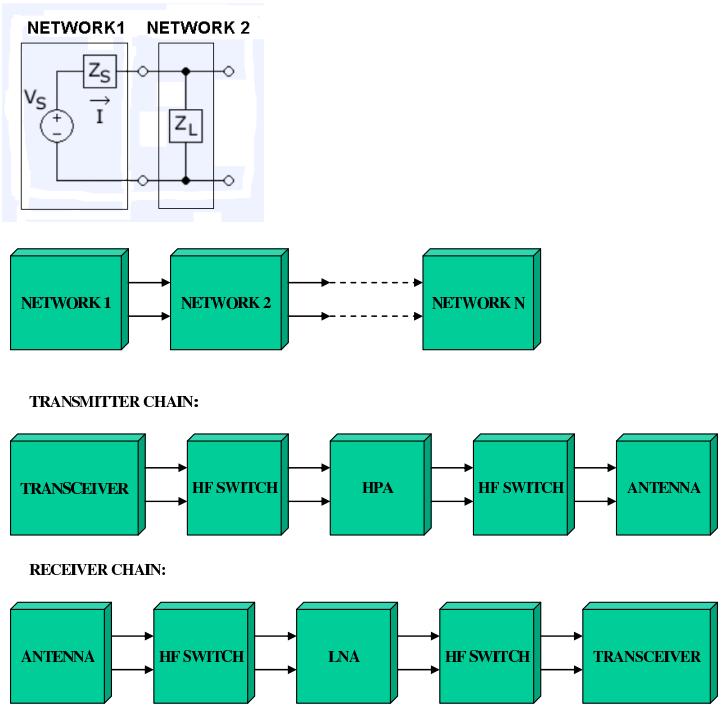

Figure 4.10: Block diagram of the 2-port network design approach

stant of the material. Calculating the maximum length (l_{max}) of the transmission lines:
$\varepsilon_r = 4.7$, (FR-4 material)
$\lambda_0 = c/frequency = 3 * 10^{-8}/437MHz = 68.7cm$

$$\lambda_{eff} = \frac{68.7cm}{\sqrt{4.7}} = 31.7cm$$
$$l_{max} = \frac{5\%}{100\%} 31.7cm = 1.58cm$$

As seen from the calculations above if the maximum length of the copper traces on the PCB is be kept below 1.58cm they do not need to be considered in a 2-port network design.

4.3 PCB

This section describes the Printed Circuit Board (PCB) design and the design methods used to realize the circuit.
The PCB is made of FR4 material and has four layers (see F), *top signal layer*, *ground layer*, *power layer* and *bottom signal layer*. The top signal layer holds all the component and RF transmission lines while the signal lines are evenly distributed on the top and bottom signal layers. The ground layer is situated below the top signal layer where it is easily accessible for ground vias. The power layer is used to freely distribute power to the various

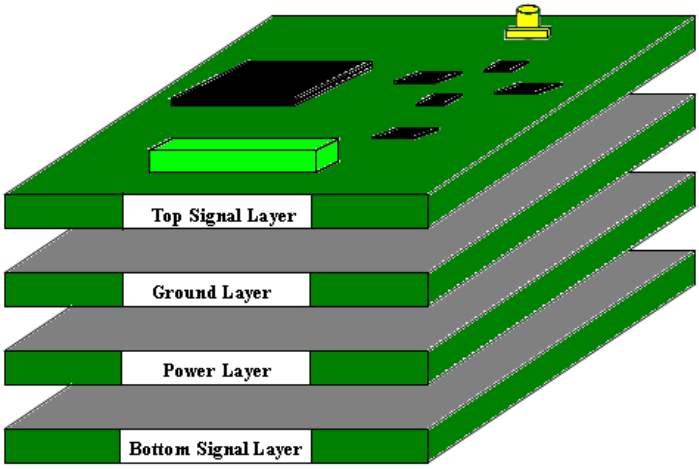

Figure 4.11: Figure of the PCB layers

components in the circuit except for the HPA and LNA which receive power from dedicated power lines from the power switches.
The components, on the top signal layer, are grouped into two physical sections (RF and digital) on the PCB to accommodate the need for a split ground plane (see subsection 4.3.2)

4.3.1 Components

In order to achieve the relative small physical dimension required in the design, only surface mounted components were used.
In the RF section of the circuit:

- all the passive components are type 0402 to reduce physical dimensions
- Resistors, 1% tolerance
- Capacitors are of the Murata GRM1556C series, recommended by Texas Instrument

- Inductors are of the Murata LQG15HS series, recommended by Texas Instrument
- all the active RF components are QFN packages for optimal thermal and electrical (parasitic capacitance and inductance) performance [18]

4.3.2 Ground Plane

The ground layer is the reference plane for the electronic circuitry. It serves as a low-impedance return path for currents and as a shield against the circuits RFI/EMC emissions and susceptibility of the same emission from nearby sources.
To maintain low impedance in a ground plane is critical, as the ground noise will increase proportionally with the ground impedance.
In mixed signal circuits (analog and digital) it is important to avoid that the digital and analog circuits share the return path in the ground plane (see figure 4.12), because digital circuits create much noise while analog circuits

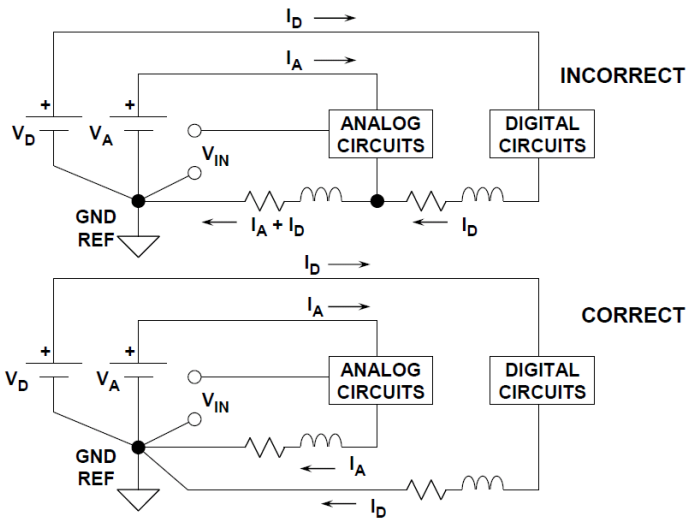

Figure 4.12: The incorrect and correct way of routing ground paths. Credited: Analog Devices

are very sensitive to noise.

There are three different design strategies which can mitigate "noisy" ground by separating the current return path [19].
The first are to route dedicated ground lines to each signal path. This technique is not popular as it complicates the design and the narrow ground tracks have higher impedance. The second are to use an unbroken ground

plane, but physically separate the analog and digital circuitry making sure the current return paths are not close to each other. The last are to split a ground plane into several planes for various type of circuitry. This is an effective design method, but somewhat limits the design possibilities as the planes must remain unbroken all the way to the common reference ground. According to the recommendation in [20] the ground layer was separated into two planes, one for digital signals and one for RF signals.

4.3.3 Decoupling Capacitors

Decoupling capacitors, often known as bypass capacitors, are used to decouple noise and transients on signal lines. It is customary to place them as close as possible to the input of the signal and to ground through an individual connection to ground (see figure 4.13). In the PCB design all active devices had decoupling capacitors close by their input voltage pins and dedicated vias to ground. Decoupling capacitors are also used on some of the long digital signal lines as well.

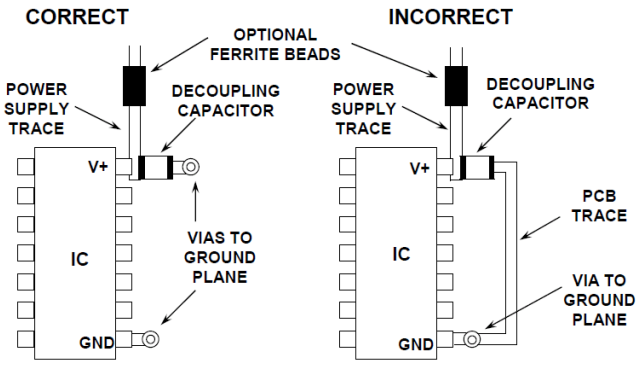

Figure 4.13: The correct and incorrect way to place decoupling capacitors. Credited: Analog Devices

Ground Vias

Ground vias is inherently small inductors and so introduce extra impedance in a current return path. Every ground connection has its own ground via to reduce the overall ground impedance for the return current and to avoid situations where several components use the same ground via and introduce ground noise (see figure 4.12 and 4.13).

4.3.4 Shielding

Shielding is used to limit the susceptibility of a circuit from the effects of Radio Frequency Interference (RFI) and Electro Magnetic Compabilty (EMC) from external sources and vice versa. As discussed in subsection 4.3.2 the ground plane under the RF section acts as a shield. In order to properly shield a circuit it must be enclosed in a EMC shield. The RF section has been enclosed by large ground vias in a rectangle which can be seen in figure 4.1.
An aluminum casing is placed above the RF section with a hole to the antenna (SMA) contact and soldered to the PCB and the vias. The PCB with the casing mounted can be seen in figure 4.14.

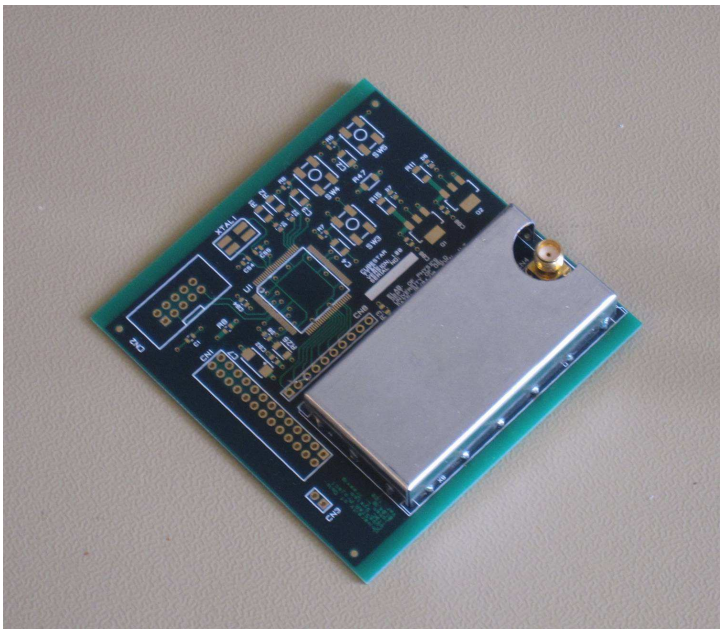

Figure 4.14: The PCB with EMC screen mounted

4.3.5 Thermal Vias

Thermal vias are placed underneath all the RF chips. The vias serve a dual purpose, (1) to provide a low impedance connection to the ground plane for each of the IC packages and (2) provide thermal relief for the LNA and HPA. Due to the small size of the amplifiers used in the RF section and the power levels they handled it was essential to add thermal vias bellow each package. The thermal vias can be seen in figure 4.15 where they have been marked with a red circle.

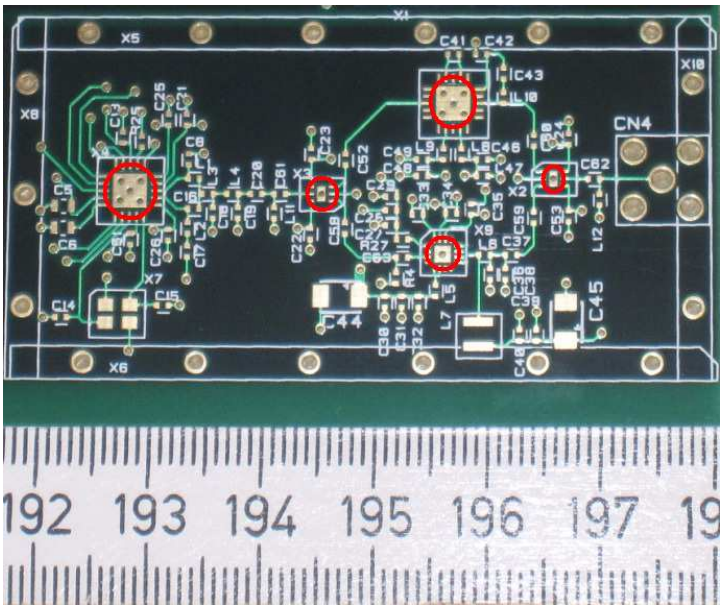

Figure 4.15: The thermal vias underneath the RF pads

4.4 Firmware

The Hardware Abstraction Layer (HAL) is a firmware library written for a particularly hardware system. In software engineering this is commonly referred to as drivers.
The purpose of the HAL is to interface the hardware and the *CubeSTAR communication protocol software*, throughout this chapter referred to as protocol layer (see [3] for more information on the CubeSTAR communication protocol). The HAL is written in C-code. From the protocol layer the HAL will make the system appear as a generic communication channel.

4.4.1 Hardware Abstraction Layer Architecture

The HAL architecture is built around a library of firmware drivers handling various hardware and firmware sections in the system and a communication module containing the necessary functions to handle a transmit or receive operation. The architecture is designed to handle transmissions of telemetry and beacon signals and reception of commands, as described in section 2.1. The *hal.h* is used to make a simple interface to the protocol layer.

Firmware Interface

The HAL packet is designed to be integrated into the CubeSTAR protocol layer, for this reason the packet was designed for an easy interface. The

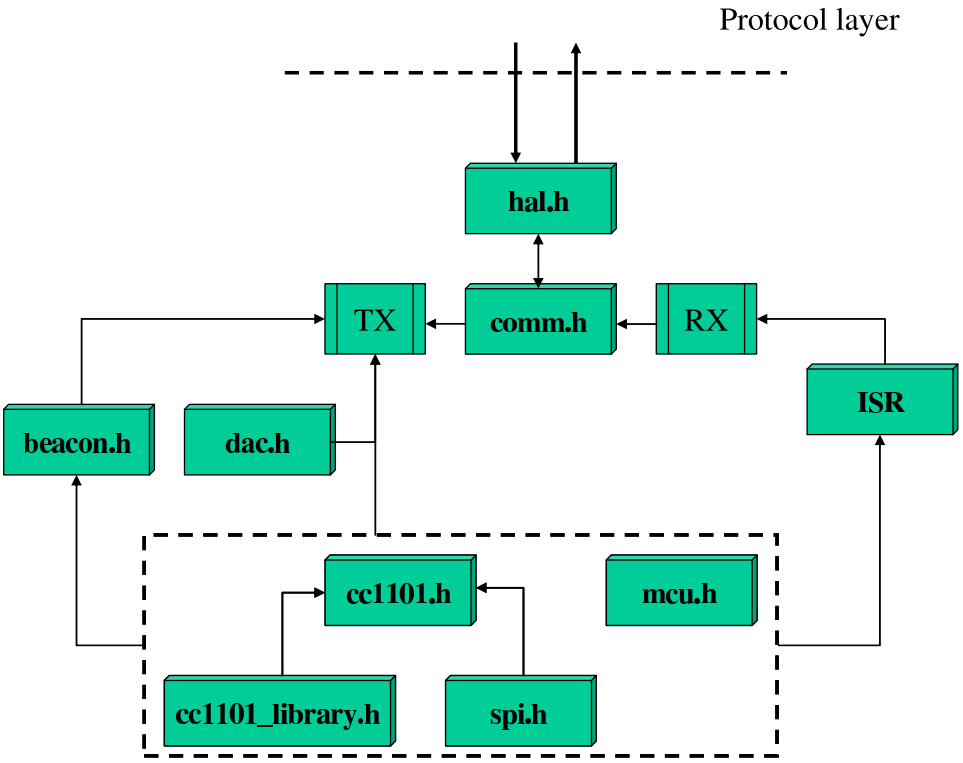

Figure 4.16: The HAL architecture

interface consists of only two functions, a bool variable and a char pointer. The *hal_ init()* is a function initializing the necessary resources to use the HAL packet. The function must be called before the protocol layer starts using the HAL packet.

Listing 4.1: The Hardware Abstraction Layer initializing function call

```
/*! \brief Calls the initialzing function call for the Hardware Abstraction Layer
*
*
* \param none
*
*    \return none
*/
#define hal_init()  (
            mcu_init(),          /* initalize the MCU resources */
            spi_init(),          /* initalize the SPI interface */
            usart_init(),          /* initalize the UART interface */
            rtc_init(),          /* initalize the Real Time Counter module */
            system_default_mode() /* place the system in default mode: Receive */
            )
```

The *send_ packet()* is used to transmit telemetry or beacon data. When the protocol layer wish to transmit either a beacon or telemetry, it will call this function specifying the transmission type, pointer to buffer and number of bytes to transmit.

Listing 4.2: The function call handling transmission of telemetry and beacon

```
/*! \brief Calls the telemetry or beacon driver
int8_t send_packet(uint8_t type_of_data, uint8_t *buffer, bytes)
*
*
* \param uint8_t type_of_data
* \param uint8_t buffer
* \param uint8_t bytes
*
*   \return Status code
*   \retval (uint8_t)OK
*   \retval (uint8_t)FAILED
*/
int8_t send_packet(uint8_t type_of_data, uint8_t *buffer, uint8_t bytes)
{
 int8_t status = OK;

 switch(type_of_data)
 {
  case(BEACON):
   send_beacon(buffer, bytes);
  break;

  case(TELEMETRY):
   send_telemetry(buffer, bytes);
  break;

  default:
   status = FAILED;
  break;
 }
 return OK;
}
```

The communication system is by default in receive mode whenever it is not busy transmitting. The system can receive data independently of the protocol layer using an interrupt routine and will indicate that a new packet is received by altering the value of *bool new_command* from false to true. The protocol layer is required to poll this variable periodically. The new data can be retrieved from the pointer *uint8_t *command_buffer*. The protocol layer is responsible for reading the command before it is overwritten by a new command.

Listing 4.3: The interrupt routine handling reception

```
/*! \brief Interrupt service routine for transceiver operations
 *
 *  When initialized this ISR is called on every rising edge
 *  on GD0.
 */
ISR(PORTE_INT0_vect)
{
 switch(trx_state)
 {
  case(TX):
   /* if transmitting finished, do nothing */
   if(!Is_GDO0_PIN_high())
   {}

  /* if transmitting started, do nothing */
   if(Is_GDO0_PIN_high())
   {}
  break;

  case(RX):

   /*  FIFO filled, download data */
   if(!Is_GDO0_PIN_high())
   {
    rx_buffer_lenght = cc1101_read_reg(CC1101_RXFIFO);
  cc1101_burst_read_reg(CC1101_RXFIFO, command_buffer, rx_buffer_lenght);
  new_command = true;
   }

   /* receiving data, do nothing */
```

```
    else
    {}
33
   break;

   /* system recovery */
   default:
38  system default mode();
   break;
  }
 }
```

HAL Hardware Resources

Since the HAL is used to control the physical layer of the communication system it requires hardware resources in the MCU.

- 1 SPI module
- 1 I2C module
- 10 generic I/O lines
- 1 interrupt vector

4.4.2 Configuring the Transceiver

The CC1101 transceiver contains 49 8-bit registers used to configure the transceiver. The manufacturer of the device, Texas Instrument has developed a software program SmartRF Studio to calculate the registers settings depending on the desired RF characteristics. SmartRF Studio was used to obtain the register values for the various configurations.

Chapter 5

Antenna

This section will discuss antenna types applicable for the CubeSTAR satellite. The reader is encouraged to read appendix C for basic understanding of antenna operations and description of the antenna types discussed in this chapter.

5.1 Introduction

In this section some of the key characteristics for antennas applicable for the CubeSTAR satellite is discussed.

5.1.1 Directional vs. Omni-directional

Because of the long distances between a satellite and the Earth and the limited power available to a satellite, it is beneficial to use antennas with a high gain.
Directional antennas like Yagi and dish antennas provide much higher gain than omni-directional antennas, but they also require a high degree of pointing control since the beamwidth is much narrower than that of an omni-directional antenna.
Generally satellites have a good Attitude Determination and Control System (ADCS) [1] which allows for use of directional antennas. Nano and pico-satellites are normally too small to use active attitude control systems like e.g. thrusters, although several interesting options are under development [21]. Most Cubesat today uses some form of magnetic actuators interacting with the Earths magnetic field to maintain a certain degree of attitude control. The CubeSTAR satellite will likely adopt this solution.
The degree of attitude control is therefore uncertain and the best option for

[1] Attitude Determination and Control System, is the system which identify the orientation of the satellite with reference to Earth and maintains a given angle towards Earth by use of passive or active systems

CubeSTAR is an omni-directional antenna as the gain is equal in all directions.

5.1.2 Polarization

All radio waves have a polarization. As described in appendix C there are *linear*, *circular* and *elliptical* polarizations.
Linear polarized waves are dependent on the orientation of the transmitting and receiving antenna. In the event the transmitting antenna is oriented vertically it will transmit vertical polarized radio waves and if the receiving antenna is oriented horizontally it can only receive horizontal polarized radio waves. In this case communication would be impossible.
If one of the antennas where circular polarized it would be possible to send radio waves back and forth, but it would still be a polarization mismatch which could cause attenuation up to 3dB of the radio signal.
In the case where both the antennas are circular polarized there would be no polarization mismatch loss except for any mismatch the atmosphere might induce [2].
The CubeSTAR ground station uses an antenna which transmits with circular polarization, so it is not critical to the communication link to use a circular polarized antenna on the CubeSTAR satellite. It is however preferable to use a circular polarized antenna to reduce the attenuation of the radio signal as much as possible.

5.1.3 Redundancy

Several factors like mechanical problems during deployment of the antenna and the environmental effects in space discussed in section 2.6, give good reasons to add redundancy to the communication system. Adding a second antenna system to the satellite increases the missions chance for success. The second antenna can be part of a back-up communication system totally independent of the primary communication system. The back-up system will be dormant and will only be powered on if the primary system experiences failure.

5.2 Antenna Configuration

The need for omni-directional antennas, simple mechanical design and the limited available space on the satellite have identified *dipole* and *turnstile* antenna as the best antenna solutions for the CubeSTAR satellite. Each of

[2] A circular polarized wave propagating through the atmosphere will experience that the phase is distorted in a way that the shape moves from circular to elliptical causing an attenuation of the signal due to polarization mismatch

the configurations presents different attributes.
Both the dipole and turnstile antenna have a near omni-directional gain which makes them ideal for a Cubesat. The major differences between the two are the physical size and the polarization.
The turnstile antenna transmits circular polarized radio waves which will eliminate any polarization mismatch loss in the radio link. This gives a better link margin and BER in the communication link.
However given the results of the link budget calculations (see section 3.3) the link margin can provide the specified BER even if the link margin drops the maximum 3dB due to polarization mismatch.
In this case it is more important to consider operational redundancy in the system. The turnstile antenna is only a crossed dipole antenna which can easily be configured to act as two separate dipole antennas. Based on this the CubeSTAR satellite should use two dipole antennas, one for a primary system and one for a back-up system. This will increase the chance of success for the satellite mission.

5.3 UiO Antenna

It was decided by the CubeSTAR project management that the team should try to develop an in-house produced antenna solution. This decision was motivated by the academic benefits of having students work with antenna design. Figure 5.1 is a sketch of the mechanical structure of the proposed antenna solution. The housing is designed to hold four quarter wave antenna

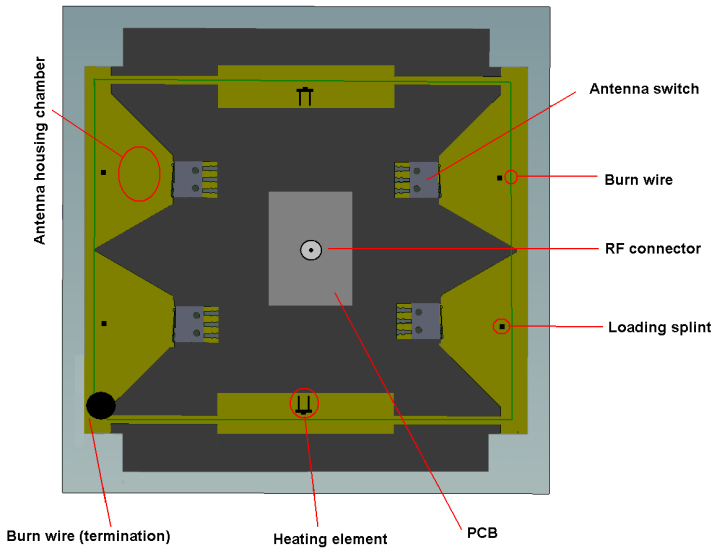

Figure 5.1: The proposed UiO antenna structure with the top cover unmounted.

elements which is connected into a x2 dipole or turnstile configuration depending on the switch matrix described later in this section. In the base of the housing chamber of each antenna element is a switch which detects if the antenna is deployed (preferably a SPDT to separate between a situation where the antenna is deployed or a failure in the switch or the electrical feed to the antenna).
The *burn wire* is placed around the antenna structure to keep the rolled up antenna elements in place before deployment. To cut the wire and deploy the antennas is a heat element used to heat the wire until it breaks. An extra heat element is added for redundancy.
The PCB is placed in the center of the mechanical structure. The circuit will contain a MCU to communicate on the CubeSTAR backplane bus, switches to change antenna configuration between x2 dipole and turnstile and a 90^o phase shifter used in the turnstile configuration. A block diagram of the electrical system for the antenna solution can be seen in figure 5.3. The

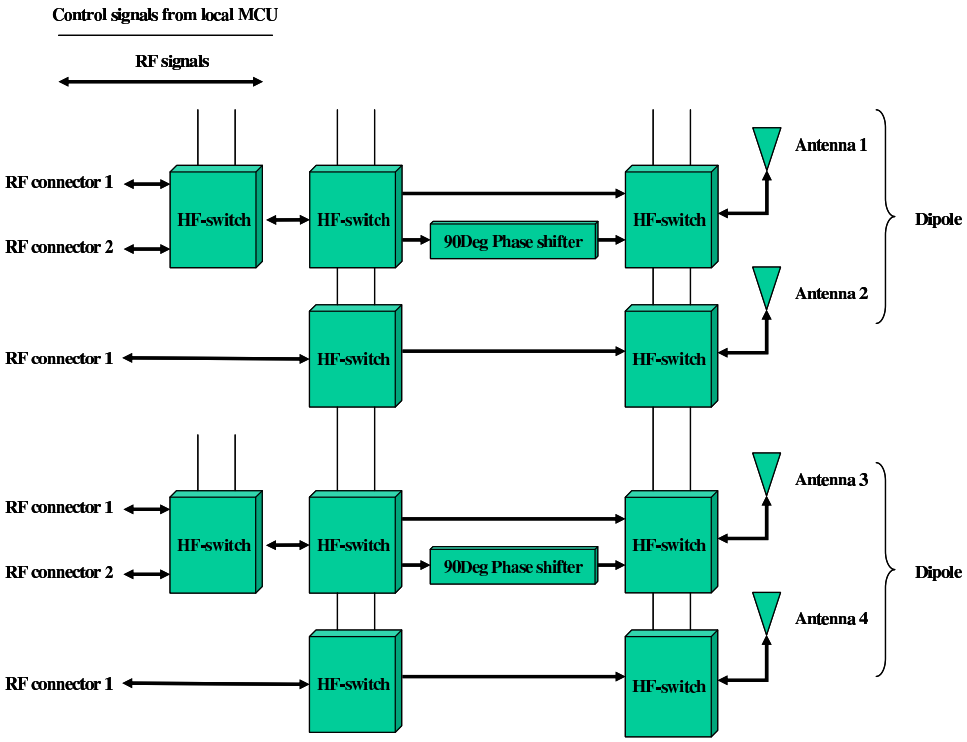

Figure 5.2: The proposed UiO antenna electrical system.

PCB will have two SMA connectors mounted on the board, they will serve as the RF input/output to the transceiver system.
In turnstile mode only one connector will be used, while in dipole mode both connectors will be used, one for each dipole. The physical placement of the antennas on the satellite structure can be seen in figure 1.1.

5.4 ISIS Turnstile Antenna

The ISIS turnstile antenna system was purchased from the Dutch company ISIS. This is a flexible antenna system for UHF transmissions and can change configuration between two dipole antennas or a turnstile antenna. This is used as a back-up alternative for the CubeSTAR satellite.

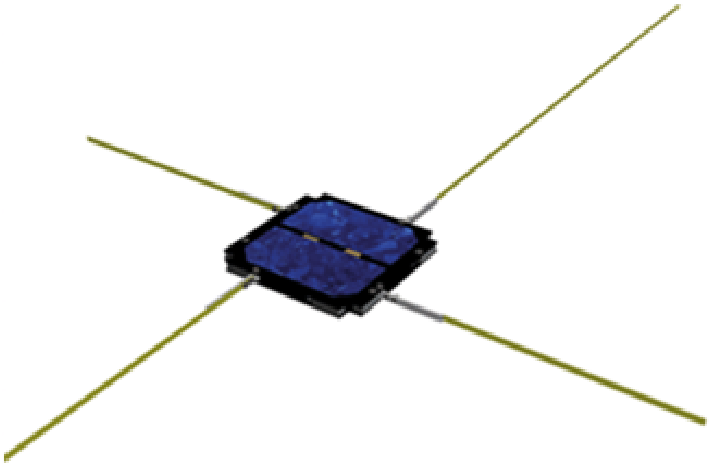

Figure 5.3: The ISIS turnstile antenna system. Credited: Cubesat.com

Chapter 6

Test and Validation of Circuit

This chapter will describe the testing and validation of the prototype system and the results generated from the tests.

6.1 Introduction

The prototype system has been developed according to the requirements discussed in chapter 2 and 3. It was however necessary to perform tests to validate the characteristics of the system.
There where several test which had to be performed:

- Test the control circuitry (digital interfaces and control signal)
- Thermal regulation of the amplifiers (LNA and HPA)
- Current consumption in various modes (Tx, Rx and idle)
- Measure Tx output power
- Establish a data link with the ground station using the AX25 protocol
- Verify a operational beacon signal

6.1.1 Test Setup

Debug and Control Interface

The debug interface was implemented in the firmware to operate and test the various section of the system. The interface uses a Universal Asynchronous Receiver/Transmitter (UART) module to communicate externally with an USB/UART bridge connected to a computer. A HyperTerminal window is used as a Graphical User Interface (GUI) to issue commands and display system telemetry.

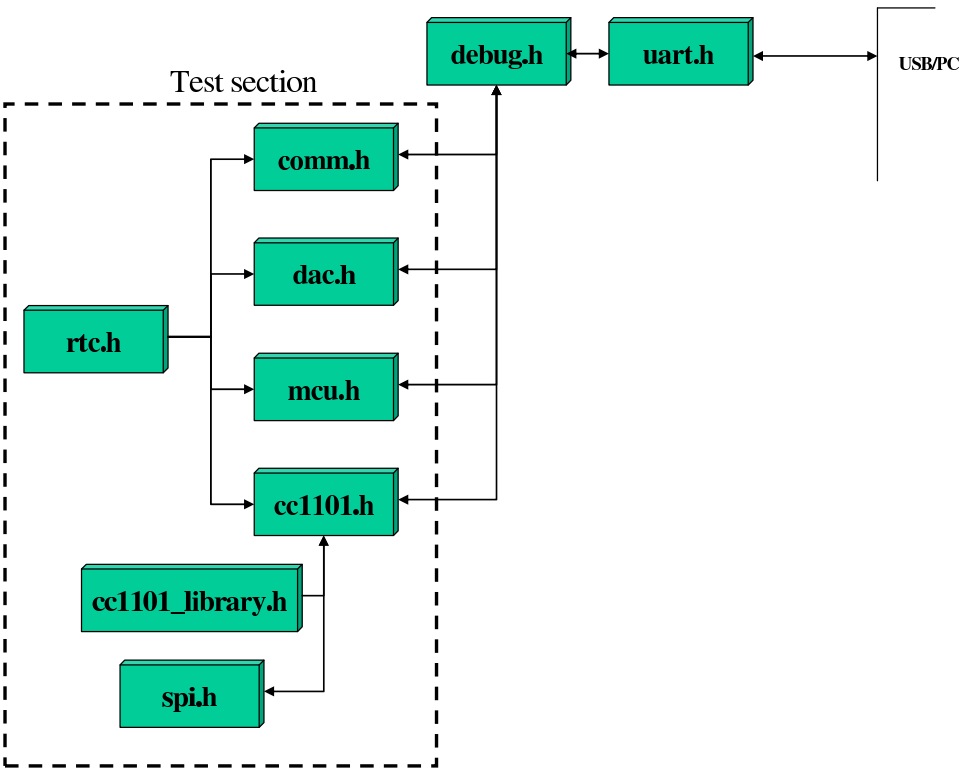

Figure 6.1: The debug firmware architecture

Test Conditions

The test setup operated in room temperature, approx. (25^o). The system supply voltage was 3V, $V_{cc} = 3.0V$. The test equipment used is listed in table 6.1.

6.2 System Design Verification

This section describes the testing performed to verify the proper electric and digital function of the circuit.

6.2.1 Test of the Control Circuitry

The first validation of the system was to verify that all the MCU interfaces (Inter-Integrated Circuit (I2C), UART and Joint Test Action Group (JTAG)) and control signal was operated correctly.

Equipment	Manufacturer	Model
SWR-meter	Sommerkamp	SK-M514
Spectrum Analyzer	HAMEG	HM5014
Calibration RF Source	Texas instrument	CC1101DK433
Digital Attenuator	Skyworks	SKY12329-350LF
50Ω RF Termination	Suhner	
Power supply	Ningbo FTZ Hopewell	PS3005
Multimeter	MASTECH	MAS830L
Cables		SMA/Coax/USB
USB/UART bridge	FTDI	TTL-232R-3V3
Termometer		
Software	ATMEL	AVR Studio 4
Software	Microsoft	HyperTerminal ver.5.1
Software	Texas Instrument	RF Studio ver.7
JTAG Programmer	ATMEL	JTAGIVE MK.II

Table 6.1: List of test equipment

Interfaces

It was crucial for the following tests that the JTAG interface (used for programming of the MCU) and UART interface (used to control and debug the circuit) were functioning properly. They were both tested for electrical connection between the MCU and their respective connector pins at the CubeSTAR back plane connector. The JTAG was tested functionally by reading the signature of the MCU from AVR Studio through the JTAGICE and later by reprogramming the MCU with updated firmware.
The UART interface was tested by writing text on the HyperTerminal window which were sent to the MCU through the USB/UART bridge cable where the data was received and sent back to the HyperTerminal.
The I2C interface was not required for testing of the system.

Control Signals

The control signals where enabled one at the time by issuing commands through the debug interface. For each signal the intended operation for each particular signal was confirmed during constant monitoring of the voltage and current consumption of the circuit.

6.2.2 Output Effect

The output effect was measured using the test setup in figure 6.2 while a continuous signal was being transmitted.
Before the test, calibration of the test setup was made using a known RF

source (CC1101DK433, see table 6.1). The scope was to determine the attenuation the test setup exerted onto the measured signal. The digital attenuator was used to protect the input of the spectrum analyzer.
The calibration of the test system showed the test system gave a total attenuation of $L_{tot} = 32.0dB$.
The measured transmitted effect for the prototype system was $P_{meas} = -2.6dBm$. By adding the loss in the test system to the measured effect the transmitted effect (P_{tx}) could be found.

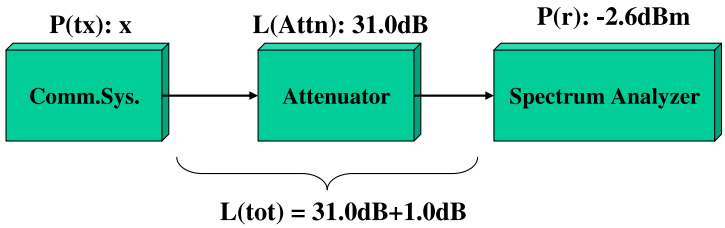

Figure 6.2: Test setup for output power measurement

$$\begin{aligned} P_{tx} &= P_{meas} + L_{tot} \\ P_{tx} &= -2.6dBm + 32.0dB \\ P_{tx} &= 29.4dBm \end{aligned}$$

The calculation above shows that the transmitted effect from the prototype system was measured to be $P_{tx} = 29.4dBm = 871mW$. This is close to the required P_{tx} defined in 3.2.1. The link margin will absorb the difference.

picture here..

6.2.3 Current Consumption

The results form the measurements of the current consumption will be important when a power budget must be calculated for the CubeSTAR satellite at a later date.
The system has three modes of operation, idle, transmitting (Tx) and receiving (Rx).
Idle mode is defined as when the MCU and transceiver (while idle) is powered. In Tx mode the MCU, the power switch, the transceiver (while transmitting), the HPA and both HF switches is powered. In Rx mode the MCU, the power switch, the transceiver (while receiving), the LNA and both HF switches is powered.
The measured values can be found in table 6.2:

6.2.4 Thermal Testing

The purpose of this subsection is to measure the ambient temperature and calculate the junction temperature using the results from subsection 6.2.3

Mode	I_{cc} **[mA]**
Idle	12
Tx	850
RX	220

Table 6.2: List of current consumption in different modes

and 6.2.2. Thermal testing was performed to verify that the thermal vias (see section 4.3.5) were able to conduct heat away from the HPA. The datasheet of the HPA specifies that the internal temperature (T_j) must be equal or below the maximum temperature ($T_{max} \leq T_j \leq 150^oC$).
The datasheet also provides a formula for calculating the T_j, based on the assumption that the ambient temperature $T_{amp} = 85^o$. However, it was not properly specified where to measure T_{amb}, so measurements where performed on both the device package and directly on the thermal vias below the device.

Measuring T_{amb}

The HPA is powered only during transmissions and will only transmit short bursts of data at the time, allowing the device to cool down between each burst. The satellite is only accessible from the ground station in time lapses less than 8 min [3] for each pass.

The HPA is tested using a continuous signal (a stream of alternating 1's and 0's) over a period of time. The device was powered on during a 2 hour period. The measured ambient temperature during the first 20 min can be seen in figure 6.3.

Calculating T_j

Equation 6.1 from the datasheet for the HPA [22] gives T_j.

$$T_j = T_{amb} + P_{diss} R_{JA} \tag{6.1}$$

where
P_{diss} is dissipated power from the device
R_{JA} is the total thermal resistance
$T_{amb} = 85^oC/W$, the surrounding temperature close to the device, according to the datasheet for the HPA

Finding P_{diss}:

$$\begin{aligned} P_{diss} &= P_{tot} - P_{tx} \\ P_{diss} &= V_{cc} * I_{cc} - P_{tx} \end{aligned}$$

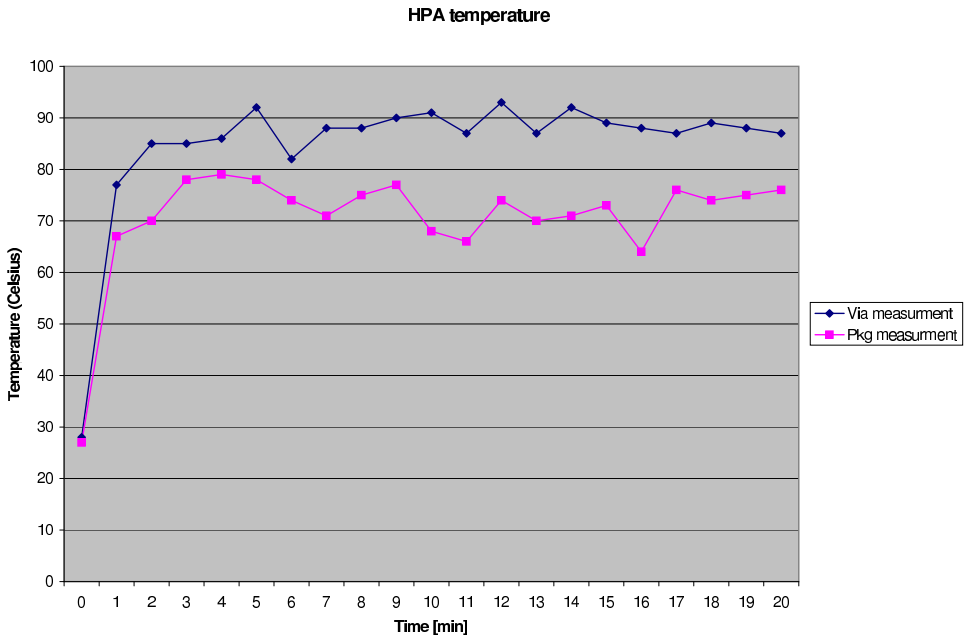

Figure 6.3: Thermal measurements on the HPA package

$$\begin{aligned} P_{diss} &= 3V * 0.850A - 1.585W \\ P_{diss} &= 0.461W \end{aligned}$$

where I_{cc} is taken from table 6.2 and P_{tx} is taken from the datasheet for the HPA

Finding R_{JA}, the thermal resistance between the junction and the ambient:

$$\begin{aligned} R_{JA} &= R_{JC} + R_{CA} \\ R_{JA} &= 25.4^oC/W + 10.6^oC/W \\ R_{JA} &= 36^oC/W \end{aligned}$$

where R_{JC} is the thermal resistance from junction to case. This value is found in the data sheet for the HPA. The R_{CA} is the thermal resistance in the PCB (R_{PCB}). This value is difficult to determine without a thermal analysis of the circuit board itself. The datasheet for the HPA presents a thermal analysis of the device mounted on a demo board. The demo board is a multilayer FR4 circuit board and has thermal vias under the package. Thus it was decided to use the thermal resistance of the demo board in this calculations.

Finding T_j (the internal temperature in the device):

$$\begin{aligned} T_j &= T_{amb} + P_{diss} R_{JA} \\ T_j &= 85^oC + 0.416W * 36^oC/W \\ T_j &= 85^oC + 15^oC \\ T_j &= 100^oC \end{aligned}$$

The calculation above shows that the junction temperature in the HPA is bellow the maximum junction temperature $T_{j(max)} \leq 150_oC$.

6.3 Systems Communication Verification

This section will describe the verification of the prototype systems communication link to the CubeSTAR ground station.

6.3.1 Establish a downlink

A downlink is defined as a communication link between a satellite and a ground station. In this case the prototype transmits a signal to the CubeSTAR ground station. The system will use a 9600 baud signal using a FSK modulated signal transmitting at a center frequency $f_c = 435MHz$. The signal is coded in AX25 packet format. The ground station uses the ICOM-910H radio and MixW software TNC to decode the received signal.

Received Signal

Several AX25 packets was transmitted from the prototype system to the ground station containing the ASCII text string: *Hello world*. Figure 6.4 shows a screenshot of the MixW software Terminal Node Controller (TNC) on the ground station while receiving the AX25 packets from the prototype system.
The signal spectrum can be seen in the lower part of the picture, where the received packets is displayed as green horizontal lines with strong signal components at f_c to the far left and the transmitted data as red/orange marks at approximately 4800Hz and with aliasing at 9600Hz. The colors in the spectrum indicates the strength of the RF component. Blue is background (thermal) noise, the green/yellow is intermittent frequency components due to the frequency shifting while red indicates a data signal.
Above the signal spectrum window is a text box which displays the data extracted from the received packets.
Transmitter>Receiver>Packet type -> CBSTAR>EARTH>U1
Data Field -> Hello world

The results displayed in the MixW software TNC indicates that the proto-

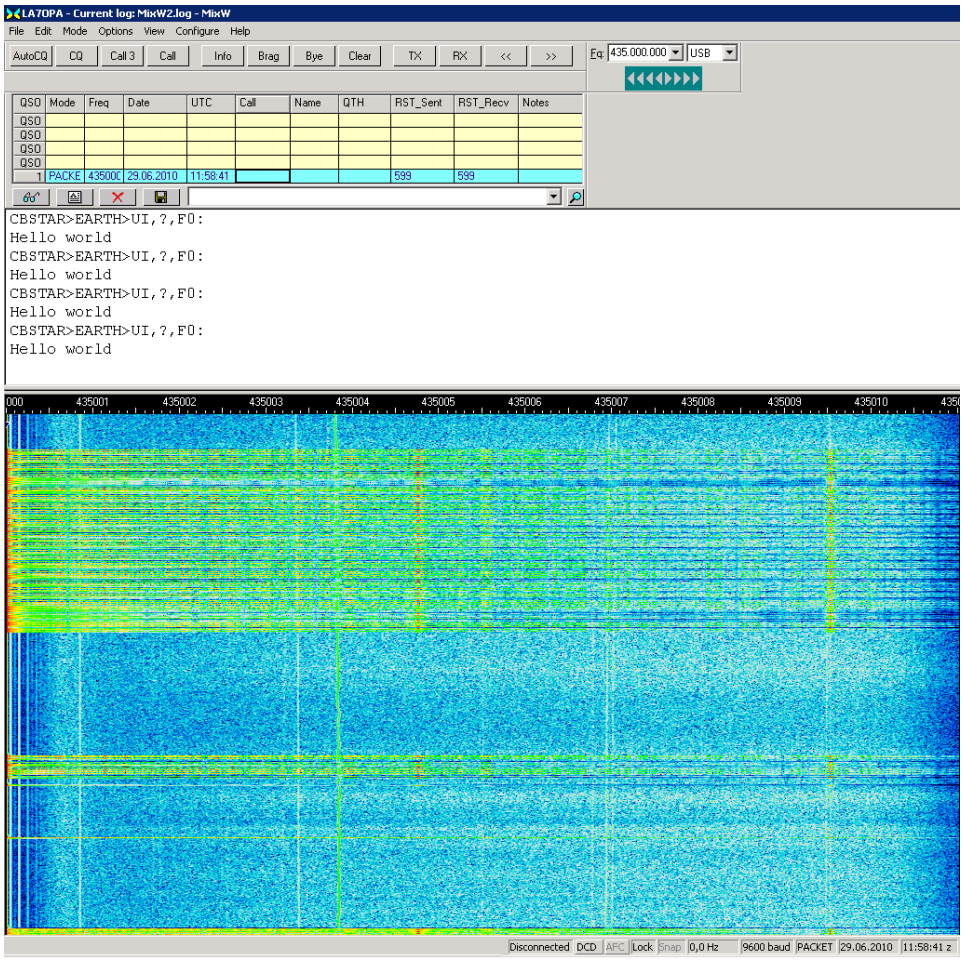

Figure 6.4: Screenshot of the MixW software TNC on the ground station receiving AX25 packets

type system can successfully transmit an AX25 packet to the ground station and maintain a telemetry downlink.

6.3.2 Establish an Uplink

An uplink is a communication link between a satellite and a ground station, where the ground station is the transmitter and the satellite is the receiver. Since the ground station is not completed it was not possible to send AX25 packets from the ground station to the prototype transceiver and verify the receiver functionality. This test must be performed at a later date.

6.3.3 Transmit a Beacon Signal

A beacon signal is as defined in subsection 2.2.6 and subsubsection 2.1 a tracking signal used to maintain optimal antenna pointing at the ground station. The signal is Morse coded and uses CW modulation scheme.

Received Signal

The beacon signal was originally defined as a CW modulated signal and it is possible to transmit a CW signal using the CC1101 transceiver, but requires complex software drivers.
Due to earlier test using a FSK modulated signals which produced a tone at the receiver end of the CubeSTAR ground station an experiment into using a FSK modulated signal to produce a beacon signal was performed instead. Utilizing the transceivers ability to transmit a preamble[1] signal, this was used to signal. The test was performed transmitting the word "PARIS" (see subsection A.3.3) to determine the Word per Minute (WPM) rate of the signal.
The ICOM-910H (ground station radio) was setup to receive a CW signal, this produced a tone each time the prototype system transmitted a signal. The figure 6.5 shows the received signal in the MixW software TNC. The colors indicate the signal strength of the signal, blue shows background and thermal noise, the green shows the frequency shifts in the signal as it shifts between zero and one. The red string is the morse coded signal. The "dots" and "dashes" can be distinguished from each other by the horizontal length of the line.
Even though a true CW signal was not used the FSK signal was able to produce an audible tone while the receiver was tuned to CW reception and the MixW TNC decoded the signal correctly. The current transceiver configuration had a frequency deviation equal to 3kHz, this can be reduced by half to emulate a more CW like signal and reduce the unwanted RF components. The results show that the prototype system can use this algorithm for beacon transmissions. The CC1101 do also have the capability to transmit a CW modulated signal, but as stated earlier in this section requires more firmware development to be implemented.

[1] Preamble, a string of binary zeroes and ones transmitted before a packet to synchronize the receiver tot he transmitter

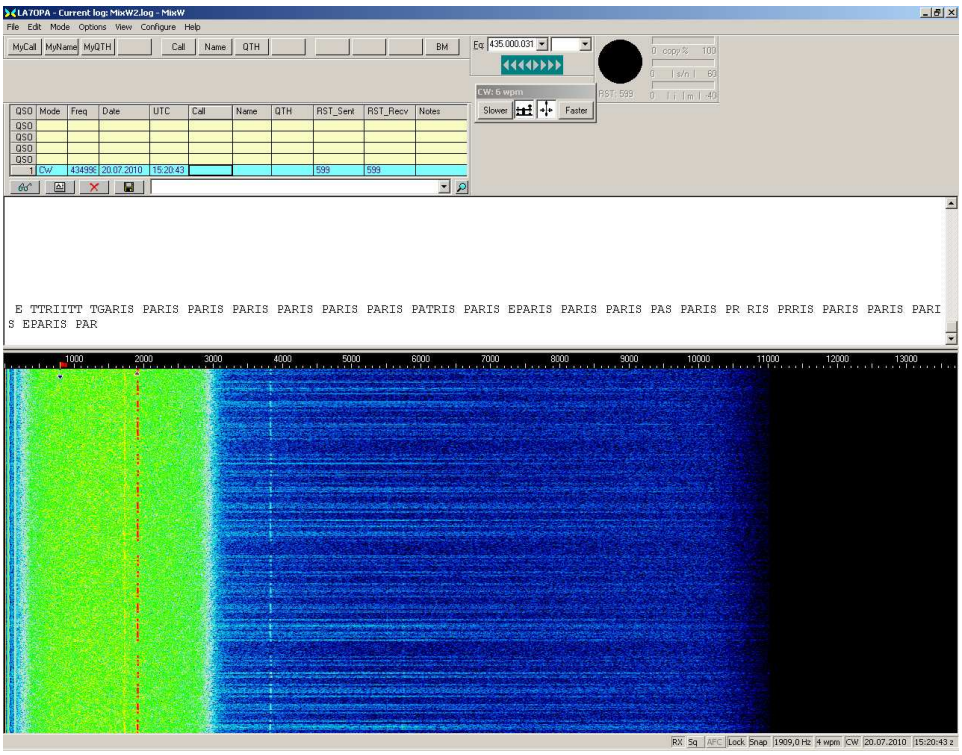

Figure 6.5: Screenshot of the MixW software TNC on the ground station receiving a Morse coded signal

Chapter 7

Discussion

7.1 Problems Encountered

This section will describe some of the major the problems encountered during the design and test phase.

7.1.1 Shift in Center Frequency

During testing of the data link (see subsection) it was discovered that the ground station radio needed to be tuned approximately 5 kHz above the selected center frequency (435MHz) to receive and demodulate the transmitted signal.
The center frequency in the transceiver is determined by the five registers FREQ1, FREQ2, FREQ3, MDMCFG0 and MDMCFG1.CHANSPC_E (see datasheet for CC1101). In order to verify that the register settings calculated by SmartRF Studio (see 4.4.2) was correct the CC1101 Evaluation kit was programmed with the same register settings as the prototype, where $f_c = 435MHz$.
Sending a signal with the evaluation kit revealed that the center frequency of the radio had to be tuned down 10 kHz in order to receive the transmitted signal. The most probable reasons for this occurrence is:

- Tuning of the register setting is necessary
- The prototype system may influence the phase of the RF signal

At present time the reason for this is not determined. Using a signal analyzer might help to analyze the signal and help conclude on the reason for the shift in center frequency.

7.1.2 PCB First Version

The system presented in this thesis is the second version developed during this thesis. The first version was produced with a wrong PCB land pattern

for the transceiver due to faulty information on the package type in the CC1101 datasheet. The transceiver chip CC1101 is a VQFN20 instead of a QFN20 as stated in the datasheet ver. E. This has been corrected in ver. F of the same datasheet.
At the time the second version was produced the digital circuitry could be verified using the already produced PCB and minor faults could be corrected. Addionally some of the LED's and switches could be removed since the debug interface had been tested. The power switches were also replaced with new types described in subsection 4.1.6.

7.2 System Limitations

This section will discuss the limitations of the prototype system.

7.2.1 Modulation Scheme

The current modulation scheme (FSK) used by the communication system is not optimal for satellite communication due to the low spectral efficiency[1] and high side-band power output.
Modern satellite communication system uses various forms of Phase Shift Keying (PSK) and special cases of FSK (see subsubsection Minimum Shift Keying) modulation scheme since they are more spectral efficent and emitts less side band power then a FSK modulated radio signal.

Minimum Shift Keying

Minimum Shift Keying (MSK) is a special type of FSK also known as Continious Phase Frequency Shift Keying (CPFSK) which share the characteristic of a continuous phase with PSK modulation due to the ratio between the signal and the frequency deviation. Normal FSK modulation scheme has discontinuities when switching between the "mark" and "space" frequency, which produces wideband frequency components. Due to the continuous phase of a MSK signal the modulation scheme will have a better spectral efficiency than standard FSK and lower side band power.
During the construction of the CubeSTAR ground station (see [23]) development into a MSK modem was started. This will be finished in the near future. Using MSK modulation scheme (with Gaussian filtering) is recommended if the current communication systems data throughput is too low.

[1]Spectral efficiency also known as spectrum efficiency and bandwidth efficiency, is a measure of how efficiently a modulation schemes can transmit bits per hertz given a limited bandwidth. Spectral efficiency is measured in bps/Hz

7.2.2 S-Band

Some amateur satellites use the S-band to communicate. The S-band allows for up to 20MHz of bandwidth which would allow for a significant increase in the data rate. This option would require further development of the ground station (antennas, LNA, etc.) and a redesign of the communication system.

7.3 Future Works

This section will discuss remaining work to complete the prototype system.

7.3.1 Software Development

Handle Packets $>$ 64 bytes

The current firmware version can transmit and receive packets smaller or equal to 64 bytes. It is expected that an AX25 packet can be larger than this and code to handle larger packets needs to be developed.

I2C Module

The CubeSTAR satellite is using an I2C bus to communicate between the respective subsystems. For the communication system to be able to communicate with the other subsystems it requires an I2C software driver.

7.3.2 EMC and RF Testing

As part of the frequency application and allocation process the communication system must pass EMC testing. The EMC testing includes characterization of unwanted emissions in the OOB and spurious domain (see 2.3.3). The European organization European Telecommunications Standards Institute (ETSI) is responsible for defining this test standards.

Test Facilities

The test and validation of RF and EMC compliance is performed using a spectrum analyzer and an anechoic test chamber which is made available by Texas Instrument Norway (former Chipcon).

7.3.3 Bandpass Filter

A bandpass filter is normally placed between the antenna connector and the rest of the circuit to attenuate frequency components outside the occupied bandwidth. Since the current prototype system has an integrated bandpass filter in the transceiver chip the external bandpass filter was omitted in this design in order to evaluate the performance of the circuit.

The transceiver circuit is built to meet the requirement of EMC and RF characteristics, however the amplification in the radio front-end section might increase the OOB and spurious emissions to unacceptable levels. The EMC and RF testing (see subsection 7.3.2) will reveal if a external bandpass filter is needed.

7.4 Recommendations

This section will describe some of the recommendations for the future development of the CubeSTAR communication system.

7.4.1 Next Version of PCB

Based on the testing of the current prototype some recommendation can be made for the next version of the PCB.

- the pull-down resistors on the control signals to the power switches must be changed to pull-ups
- the power switches must be relocated to the base of the RF section to meet the requirements for a split ground plane
- a bandpass filter must be added in the RF signal path between the antenna connector and the first HF-switch
- change the HPA gain control from the current DAC control to a fixed voltage (2.8-2.9V) using fixed resistors
- cancel the user interface (LED's and switch)
- add an extra power switch at the power line before the rest of the circuitry except for the MCU to protect the circuit (see figure 7.1)

7.4.2 System Redundancy

The benefits of adding a secondary transceiver system to the communication system has been discussed in both subsubsection 2.6.1 and subsection 5.2. The possibility of including two transceiver systems on the same PCB should be explored.

7.4.3 Adaptive Radio

Downloading telemetry from the satellite is possibly the most power consuming process on the CubeSTAR satellite. The system is designed to transmit with a signal power strong enough to maintain a $S/N \geq 10 - 12dB$ when the satellite is at maximum distance (approx. 1900km), however this distance

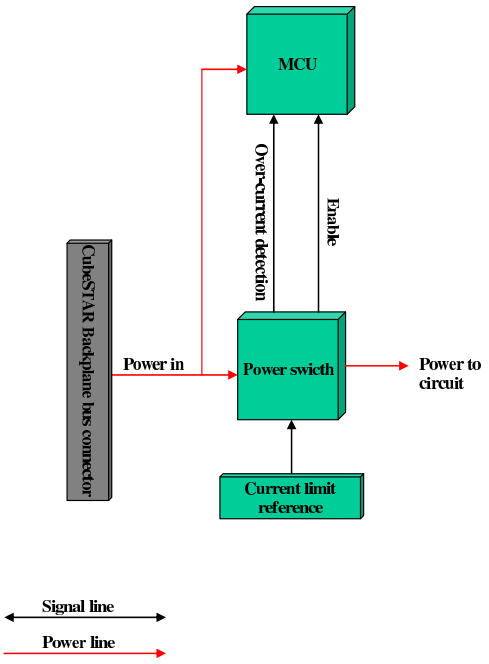

Figure 7.1: An extra power switch added to protect the circuit

will decrease rapidly from Aqusition Of Signal (AOS)[2] to nadir[3] and increase again from nadir to Loss Of Signal (LOS)[4]. This may allow for a dynamic signal power output from the system while still being able to maintain the specified link margin.
A study into implementing an algorithm to account for the variable distance should be performed. The algorithm may use the built in functions, Received Signal Strenght Indicator (RSSI) and Link Qualtiy Indicator (LQI), in the transceiver chip to measure the strength and the quality of the signal from the ground station and in doing so determine the required RF output power from the system.
The algorithm may also obtain parameters from the ADCS system to include the orientation of the satellites antenna to account for the antenna gain in the calculations.
An effective algorithm could conserve the satellites power.

[2]AOS is when the ground station starts receiving signals from the satellite when it first appears over the horizon

[3] Nadir is the position when the satellite is directly above the ground station

[4]LOS, is when the satellite disappears over the horizon and the ground station losses the signal from the satellite

Chapter 8

Conclusion

This chapter will summarize the work and the results found in this thesis.

8.1 The Prototype Communication System

In this thesis a semi-duplex radio transceiver operating in the UHF amateur satellite frequency band (434.79-438MHz) has been defined and implemented.
The system is compatible to the GENSO, transmitting at 4800 baud using a GFSK modulation scheme. Through testing the system has been verified to uphold the electrical, thermal, regulatory and functional requirements discussed in chapter two and three. Testing the EMC characteristics of the system remains.
A two dipole antenna configuration has been identified as the most suitable solution for the CubeSTAR satellite. This is based on factors like attitude control onboard the satellite, calculated link budget margin and operational redundancy.
The HAL firmware packet has been designed and operates as a interface between the protocol layer and the hardware, the packet has not been integrated into the protocol layer, but has a well defined software interface.
The prototype system has been tested against the CubeSTAR ground station and the compability has been confirmed as much as possible. Both a AX25 data packet and a beacon signal has been transmitted successfully from the prototype system to the CubeSTAR ground station. Sending a command from the ground station to the prototype system is not yet possible due to remaining work at the ground station.

Bibliography

[1] J. Antonsen, T. Houge, THE NORWEGIAN STUDENT SATELLITE PROGRAM, ANSAT

[2] T.A. Bekkeng et al., Design of a multi-needle Langmuir probe system

[3] M. Grønstad, Implementation of a communication protocol for CubeSTAR

[4] GENSO Homepage
http://www.genso.org/index.php?option=com_content&view=article&id=1:what-is-genso&catid=1:about-genso&Itemid=3
Accessed: 12 Oct. 2009

[5] K. Voormansik, Satellite Signal Strength Measurements with the International Space University Ground Station and the University of Tartu Ground Station

[6] AX.25 Link Access Protocol for Amateur Packet Radio, Version 2-2, Rev. July 1998

[7] Norwegian Post and Telecommunication Authority, The Norwegian Frequency Allocation Map
http://www.npt.no/iKnowBase/Content/Frequency_allocations_NORWAY.pdf?documentID=49095
Accessed: 15 Feb. 2010

[8] The Regulation of Radio Amateur License in Norway
http://www.lovdata.no/ltavd1/filer/sf-20091105-1340.html
Accessed: 25. June 2010

[9] The QB50 project
http://www.vki.ac.be/QB50/download/workshop/papers_18nov/beavis.pdf,
Accessed: 15. Apr. 2010

[10] International Telecommunication Union, Recommendation ITU-R SM.1138
http://life.itu.int/radioclub/rr/frr.htm
Accessed: 16. May 2010

[11] International Telecommunication Union, Radio Regulations Vol. 1-4
http://life.itu.int/radioclub/rr/frr.htm
Accessed: 15 Mar. 2010

[12] ELAB, CubeStar & ELAB,
http://www.fys.uio.no/omfi/enhetene/elab/CubeStar_and_ELAB.htm

[13] Swisscube Frame-Structure and Thermal control
http://swisscube.epfl.ch/pdf/Frame.pdf
Accessed: 15. Mar. 2010

[14] L. Jacques, Thermal Design of the Oufti-1 nanosatellite

[15] B. Klofas, J. Anderson, A Survey of CubeSat Communication Systems, Table 1

[16] Gordon, G. D., & Morgan, W. L., Principles of Communications Satellites, p.150, Pub.1993

[17] L.Besser & R.Gilmore, Practical RF Circuit Design for Modern Wireless Systems Volume 1, p.41

[18] Texas Instrument, QFN Application Report
http://www.ti.com/lit/scba017
Accessed: 15. May 2010

[19] T. Williams, EMC for Product Designers, Second Edition, 1996, p.156

[20] Analog Devices, Grounding Data Converters and Solving the Mystery of "AGND" and "DGND"

[21] J. Mueller, J. Ziemer, R. Hofer, R. Wirz, and T. O'Donnell, A Survey of Micro-Thrust Propulsion Options for Microspacecraft and Formation Flying Missions

[22] Datasheet of RF5110G High-Power Amplifier

[23] H. Vangli, Construction of a remotely operated satellite ground station for low earth orbit communication

[24] J.Clerk Maxwell, A Dynamical Theory of the Electromagnetic Field

[25] T. Pratt, C. Bostian, J. Allnut, Satellite Communications, Second edition, p.106

[26] L.Besser & R.Gilmore, Practical RF Circuit Design for Modern Wireless Systems Volume 1, p.36

[27] C. Noe, Design and Implementation of the Communications Subsystem for the Cal Poly CP2 Cubesat Project

List of Figures

List of Tables

Acronyms

ADCS Attitude Determination and Control System

AFSK Audio Shift Keying

AMSAT Amateur Satellite Union

ANSAT Norwegian Student Satellite Program

AOS Aqusition Of Signal

ARR Andoya Rocket Range

ASK Amplitude Shift Keying

BER Bit Error Rate

BPS Bit Per Second

Calpoly California Polytechnic University

CDMA Code Division Multiple Access

CEPT European Conference on Postel and Telecommunication Administration

CPFSK Continious Phase Frequency Shift Keying

CW Continuous Wave

ECC Electronic Communication Comittee

EIRP Effective Isotropic Radiated Power

ELAB the University of Oslo's Electronic Workshop

EMC Electro Magnetic Compabilty

EMI Electro Magnetic interference

EPS Electrical Power System

ETSI European Telecommunications Standards Institute

FDMA Frequency Division Multiple Access

FM Frequency Modulation

FSK Frequency Shift Keying

FSPL Free-Space Path Loss

GENSO Global Educational Network for Satellite Operation

GFSK Gaussian Frequency Shift Keying

GS Ground Station

GSN Ground Station Network

GUI Graphical User Interface

HAL Hardware Abstraction Layer

HPA High Power Amplifier

I2C Inter-Integrated Circuit

IARU International Amateur Radio Union

IL Insertion Loss

ITU International Telecommunication Union

ITU-R International Telecommunication Union - Radiocommunication Sector

ITU-RR International Telecommunication Union- Radio Regulations

JTAG Joint Test Action Group

LEO Low Earth Orbit

LNA Low Noise Amplifier

LOS Loss Of Signal

LQI Link Qualtiy Indicator

LSB Lower Side Band

MC Mission Control

MCU MicroController Unit

m-NLP multiple Neddle Langmuir Probe

MSK Minimum Shift Keying

NAROM Norwegian Centre for Space-related Education

NPT Norwegian Post and Telecommunication Authority

NSC Norwegian Space Center

OBDH On Board Data Handler

OOB Out-of-Band

PCB Printed Circuit Board

PSK Phase Shift Keying

OOK On Off Keying

RAAN Right Ascension of the Ascending Node

RL Return Loss

RF Radio Frequency

RFI Radio Frequency Interference

RSSI Received Signal Strenght Indicator

SC Space Craft

SEL Single Event Latchup

SEU Single Event Upset

SNR Signal to Noise Ratio

TBD To Be Determined

TDMA Time Division Multiple Access

TID Total Ionization Dose

TMR Tripple Modular Redundancy

TNC Terminal Node Controller

VHF Very High Frequency

UART Universal Asynchronous Receiver/Transmitter

UHF Ultra High Freqency

UiO University of Oslo

WPM Word per Minute

Appendix A

Satellite Communication Theory

This appendix contains basic satellite communication theory linked to chapter 2 and 3.

Satellite communication is a method of using artificial satellites to relay information from on spot on the Earth to another. Satellites play a major part in modern communication systems, and is used for telecommunication, broadcasting services and navigation.

A.1 Orbital Mechanics

This section will describe the fundamental laws of satellites orbiting the Earth.

A.1.1 Kepler's Laws

Kepler's laws describe the motion of two objects orbiting each other. They where postulated by the German astronomer Johannes Kepler in the 17th century. There are three laws[25]:

- The orbit of any smaller body about a larger body is always an ellipsis, with the center of mass of the larger body as one of the two foci
- The orbit of the smaller object sweeps out equal areas in equal time (see fig.A.1)
- The square of the period of revolution of the smaller body about the larger body equals a constant multiplied with a third power of the semi major axis of the orbital ellipse. That it, $T^2 = (4\pi^2 a^3)/\mu$ where T is the is the orbital period, a is the semi major axis of the orbital ellipse a μ is the Kepler constant.

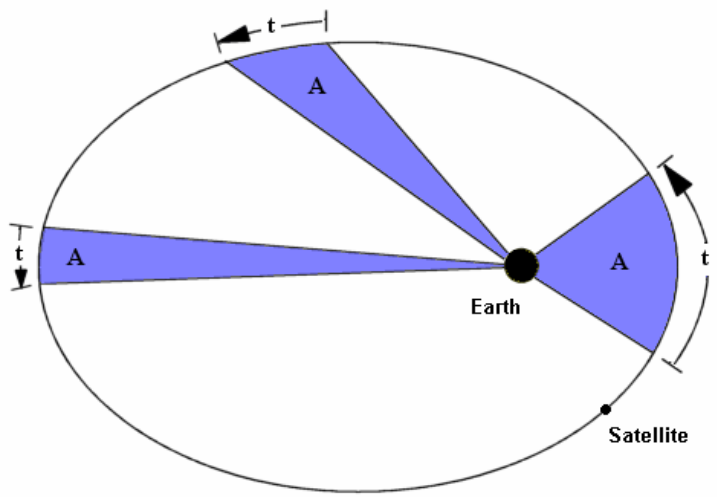

Figure A.1: Kepler's second law

Kepler's laws are used to determine the motion of a satellite orbiting the Earth.

A.1.2 Orbital Parameters

Orbital parameters is used define an unique orbit and position of a satellite in reference to Earth. Orbital elements are often referred to as *Experian elements* since they are based on Kepler's laws (see section A.1.1). There are six orbital elements (see figure fig:orb-parameters).

Eccentricity

Eccentricity (e) describes the relationship between the semi-major and semi-minor axis in the orbit. In the event that the orbit is circular the eccentricity is zero.

Semi-Major Axis

The semi-major axis (a) defines the distance between the satellite and the Earth when the satellite is closest to Earth, this is also known as *perigee*.

Right Ascension of the Ascending Node

Right Ascension of the Ascending Node (RAAN) defines the longitude where the satellite crosses from the southern hemisphere into the northern hemisphere, this point is also known as the *ascending node* (*Omega*). RAAN (Ω)is given by the angle between a reference axis in the equatorial plane known as the *Vernal equinox* and the ascending node.

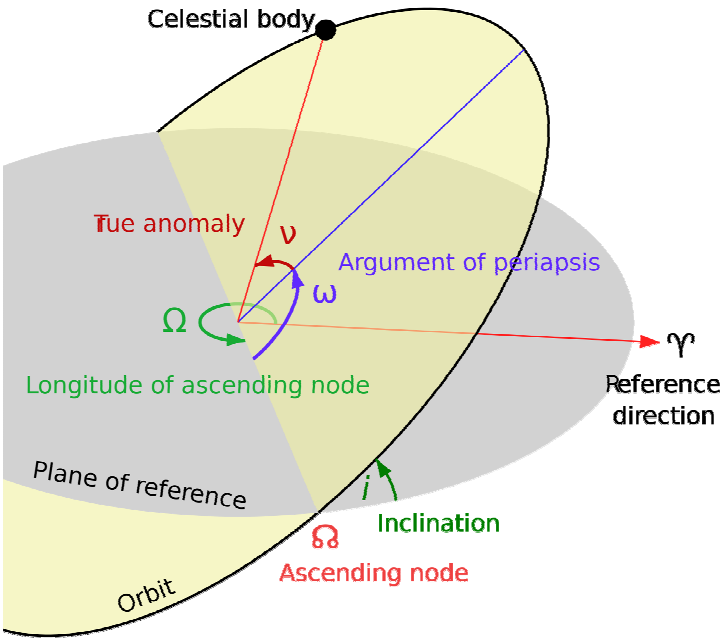

Figure A.2: Orbital parameters. Credited: Wikipedia

Inclination

Inclination (i) defines the angle between the equatorial plane and the orbital plane in the ascending node.

Argument of Perigee

The argument of perigee describes the angle between the ascending node and the semi-major axis or perigee.

True Anomaly

The *true anomaly* (ν) is a parameter defining the position of the satellite in its orbit. The true anomaly is given as an angle between the perigee and the satellites current position.

A.2 Transmission Theory

A.2.1 EIRP

EIRP is the emitted effect from a transmitter. EIRP is the sum of the delivered effect to the antenna P_t and the gain of the antenna G_t

$$EIRP = P_t * G_t \tag{A.1}$$

A.2.2 Flux Density

Flux density is a measure of the effect [W] per m^2. The energy emitted from an antenna propagates outwards in a spherical shape. As the distance from the antenna increases the sphere increases and the flux density decreases. Flux density F is given by eq. A.2:

$$F = \frac{P_t G_t}{4\pi r^2} \tag{A.2}$$

where
P_t is the transmitted effect
r is the distance the energy has propagated

A.2.3 Received Effect

The effect received by the receiver (P_r) is given by the flux density and the areal of the receiving antenna.

$$P_r = F * A \tag{A.3}$$

In practical antennas some of the energy will be reflected and some will be lost due to lossy media. The efficacy coefficient (e) is introduced to account for these losses (see subsection C.3.4). The effective areal is given by A_e:

$$A_e = e * A \tag{A.4}$$

Substituting for eq.A.5 the new equation for received effect is:

$$P_r = F * A_e \tag{A.5}$$

The Receiving Antenna

The gain of a receiving antenna is given by eq.A.6

$$G = \frac{4\pi A_e}{\lambda^2} \tag{A.6}$$

From the equation it can be stated that the gain of an antenna is influenced by both the area of the antenna and the wavelength of the signal.

A.2.4 Friis Equation

The *Friis equation* also known as the *Link equation* is the basic formula for calculating the received effect in any radio link. Substituting for A_e and F in eq.A.5 gives the formula:

$$P_r = \frac{P_t G_t G_r}{(4\pi r/\lambda)^2} \tag{A.7}$$

This can also be written in a more generic form:

$$P_r = \frac{EIRP * Receiving\ antenna\ gain}{Path\ loss} \tag{A.8}$$

A.2.5 Free Space Path Loss

The free space path loss L_{FSPL} is the loss of power to a RF signal due to the distance it must travel. From eq.A.8 and A.7 the path loss is rewritten as:

$$L_{FSPL} = \left(\frac{4\pi r}{\lambda}\right)^2 \tag{A.9}$$

where
r is the maximum distance between the receiver and transmitter (see *Slant range*).

Slant Range

Slant range is the maximum distance between a ground station and a satellite. The slant range is defined by the height of the orbit of the satellite and the minimum elevation of the ground station antenna. The slant range can be calculated using eq.A.10:

$$d_{max} = \sqrt{(R+h)^2 - R^2\cos^2\theta} - R\sin\theta \tag{A.10}$$

Where
R = is the Earth's radius ($R = 6378km$)
h = the satellites height above the Earth's surface
θ = the elevation angle for the ground station antenna

A.2.6 Noise

Electrical noise is defined as all electrical energy within the passband of a signal that is not part of the originally transmitted signal. The noise can consist of a number of frequencies and amplitudes that may affect the quality of the signal. Noise is divided into two categories, *uncorrelated* and *correlated* noise.

Uncorrelated Noise

Uncorrelated noise is noise that is present in a communication system regardless of whether a signal is transmitted or not. This type of noise can be generated by both *external* and *internal* sources.

External noise sources are primarily generated from:

- Atmospheric noise, naturally occurring electrical disturbance in the atmosphere.

- Extraterrestrial noise, energy radiated from the Sun and cosmic radiation

- Man-made noise also known as *industrial noise*, radio waves generated by electrical equipment (interference from other communication systems, spark plugs, generators, etc.)

Internal noise is generated from three sources:

- Shot noise is the noise caused by random carriers (electrons and holes) arriving at the output of a device

- Transit-time noise is caused by any change of a stream of carriers as they pass through a device and cause an irregular variation.

- Thermal noise, see below.

Correlated Noise

Correlated noise is noise created internally whenever a signal is present. The noise is generally created by imperfect components which generate nonlinear distortion such as *harmonics distortion* and *intermodulation*.

Thermal Noise

Thermal noise. *Thermal noise* also known as *Brownian noise*, *Johnson noise*, *Nyquist noise* or *white noise* is caused by the rapid random movement of electrons due to thermal effects. The movement of a free electron equals a small pulse of current which generates an AC component due to the internal resistance in the conductor. Thermal noise is evenly distributed over the full frequency spectrum and present in all electronic equipment and systems.
Thermal noise power is proportional to the bandwidth and temperature and is given by equation A.11

$$P_n = kT_nB_n \tag{A.11}$$

where
k = Boltzmann's constant = $1.39 \times 10^{-23} J/K = -228.6 dBW/K/Hz$
T_n[K] = the physical temperature of the object in Kelvin
B_n[Hz] = noise bandwidth. The bandwidth of the system, B_{-3dB}[25], is often used because B_n is commonly unknown.

Noise Factor

The noise factor is a measure of the degradation of the S/N ratio as the signal passes through a circuit.
The noise figure (NF) is given by eq.A.12:

$$NF = \frac{S/N_{in}}{S/N_{out}} \tag{A.12}$$

A.2.7 Doppler Shift

Doppler shift is a phenomenon where a constant frequency is changed because the receiver is moving relative to the transmitter. The change in frequency is proportional to the relative speed between the receiver and the transmitter. The frequency will increase when the receiver is moving towards the transmitter and decrease when moving away.
A common analogy is a train sounding the whistle, when moving towards you the sound of the whistle will be more high pitched than when the train moves away.
Doppler shift is explained in equ. A.13

$$f_r = (1 + \frac{v}{c})f_t \tag{A.13}$$

where:
f_r = received frequency
v = the speed of the transmitter relative to the receiver
c = light speed
f_t = transmitted frequency

$$\Delta f = f_t \frac{v}{c} = f_r - f_t \tag{A.14}$$

where:
Δf = frequency deviation or Doppler shift

A.3 Modulation Scheme

A.3.1 Frequency Shift Keying

Frequency Shift Keying (FSK) is a digital Frequency Modulation (FM) modulation scheme, see figure A.3. A digital symbol is modulated into a carrier wave by decreasing or increasing the carrier frequency (f_c) by a fixed frequency deviation (Δf), making $'1' = f_{mark} = f_c + \Delta f$ and $'0' = f_{space} = f_c - \Delta f$.

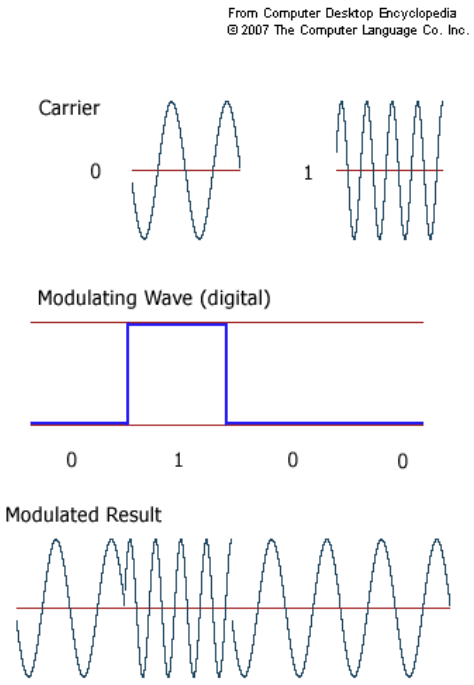

Figure A.3: Frequency Shift Keying

Bandwidth

The bandwidth B of a FSK modulated signal is calculated using *Carsons rule*:

$$B = 2(\Delta f + \frac{baud}{2}) \tag{A.15}$$

where
Δf = frequency deviation
$baud$ = signal rate

A.3.2 Audio Frequency Shift Keying

AFSK is a special FSK modulation scheme using two tones to modulate a binary signal into a carrier wave. This scheme has been popular in radio and telephone communication since it uses audible tones which can be transmitted through electronic circuits carrying sound. The usage of the first early modems to transmit data through the phone line is an example of this.

A.3.3 Morse Code

Morse code is a simple communication protocol used to transmit letters and numbers in textual form. The protocol uses a modulation scheme called

On-Off Keying (OOK) to transmit information. OOK is a basic form of Amplitude Shift Keying (ASK) where a continuous signal (e.g. carrier wave) is turned on and off (see figure A.4).
Morse code uses two signals, "dot" (•) and "dash" −, a specific timing scheme is used to separate signals, letters and words:

- a "dash" is three times as long as a "dot"
- the delay between two signals in the same letter is a "dot"
- the delay between each letter is three "dots"
- the delay between a word is seven "dots"

Morse code uses the term (Word per minute) WPM to quantify the data rate, text books in communication theory suggest two ways to determine the WPM of a morse code signal.
The first approach is to measure how many times the word "PARIS" can be transmitted in one minute. The second is to divide 1.2s with the time of one "dot", $WPM = 1.2s/T_{dot}$.

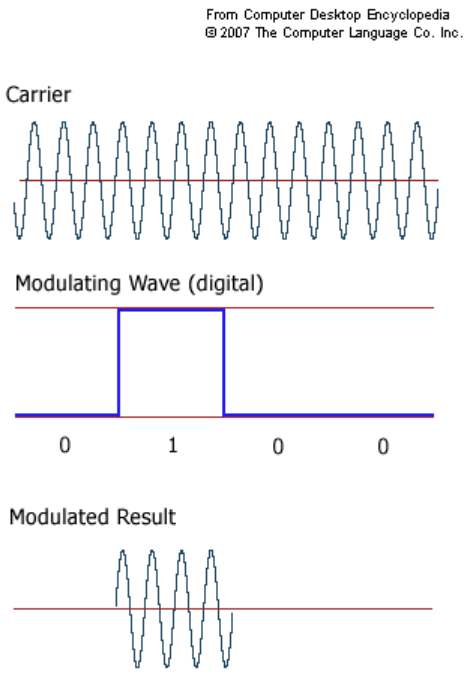

Figure A.4: ASK modulation, OOK scheme

Appendix B

RF Design Methods

This chapter will describe the methods and techniques used to implement the RF design. In RF design conventional design methods using lumped element models do not apply because the circuit will behave different at high frequencies. The characteristics of components will change value dependent on the frequency, signal lines become complex impedances and signal energy is reflected from the input on electronic circuits.
This effects forces RF designers to use the high frequency models and other design techniques described in this chapter.

B.1 Optimal Power Transfer

In RF design, the most important principle is *optimal power transfer*, which means that all the power is transferred from a source to a load. RF design theory shows that if the impedance of a load (Z_L) is the complex conjugate value of the impedance of a source (Z_S), all the power from the source is transferred to the load [26].
An impedance is always a complex number (a real part and an imaginary part), thus the complex conjugate of an impedance $Z = R + jX\Omega$:

$$Z_S = R + jX\Omega$$
$$Z_L =^* Z_S = R - jX\Omega$$

where
R = resistance
X = reactance, [+jX = L (inductive), -jX = C (capacitive)]

To simplify the design process the term *characteristic impedance* is introduced.

B.2 Characteristic Impedance

Characteristic impedance (Z_0) is an impedance where the imaginary part is equal to zero. This makes the impedance purely resistive, $Z_S = Z_L = Z_0 = R$. If all the ports in a design have an impedance equal to the characteristic impedance the design of impedance matching circuits is greatly simplified. The *de facto* reference value in conventional RF circuits is 50Ω. Even if the

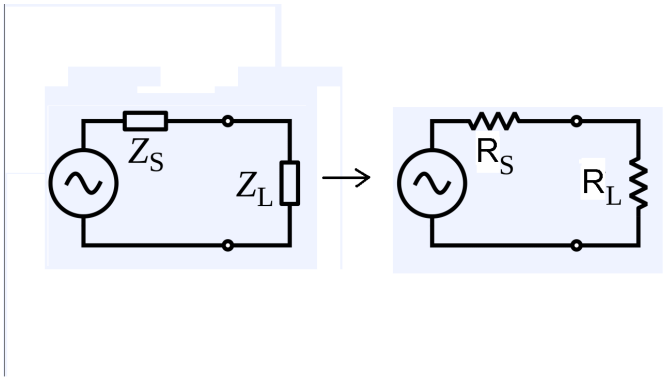

Figure B.1: Applying characteristic impedance to a source and a load

principles of optimal power transfer and characteristic impedance are implemented in a RF application there will always be some degree of attenuation to a RF signal when transferred into a load. This is caused by *return loss* and *insertion loss*.

B.3 2-Port Network

A 2-port network design is a useful design strategy when designing a RF circuit. By defining various sub-circuits as ports with two inputs and two outputs the RF circuit can be regarded as a cascading series of ports and by applying the characteristic impedance principle between each port a large number of ports can be added to a RF signal chain with minimum signal loss. A 2-port network is described through S-parameters.

B.3.1 S-parameters

Scattering parameters (S-parameters), are variables which describe the behavior of a "black body" in terms of power waves. These parameters are widely used in RF design to describe single and multiport networks. By adding the S-parameters of two ports connected to each other trough matrix algebra the calculations will produce a new set of S-parameters describing the two networks as a single network.

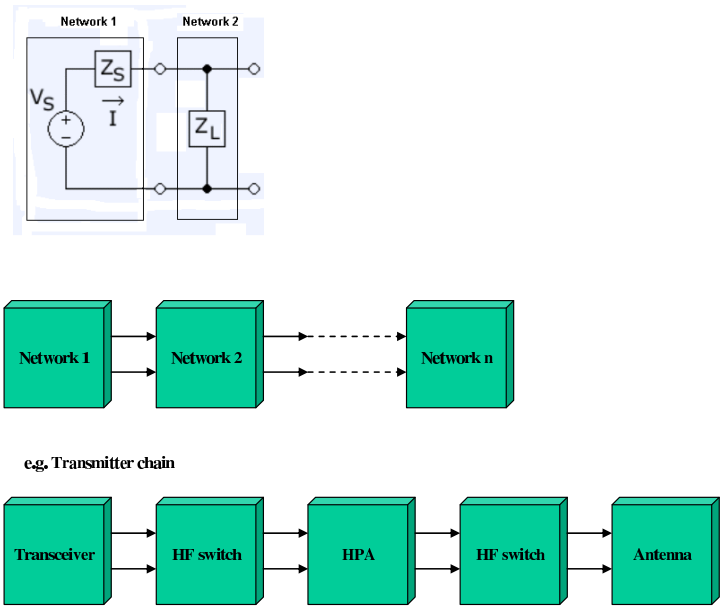

Figure B.2: Block diagram of two-port network principle

There are four S-parameters describing a 2-port network:

$$S_{11} = \frac{b_1}{a_1} \mid_{a_2=0} = \frac{reflected\,power\,wave\,at\,port\,1}{incident\,power\,wave\,at\,port\,1} \tag{B.1}$$

$$S_{12} = \frac{b_1}{a_2} \mid_{a_2=0} = \frac{transmitted\,power\,wave\,at\,port\,1}{incident\,power\,wave\,at\,port\,2} \tag{B.2}$$

$$S_{21} = \frac{b_2}{a_1} \mid_{a_2=0} = \frac{transmitted\,power\,wave\,at\,port\,2}{incident\,power\,wave\,at\,port\,1} \tag{B.3}$$

$$S_{22} = \frac{b_2}{a_2} \mid_{a_2=0} = \frac{reflected\,power\,wave\,at\,port\,2}{incident\,power\,wave\,at\,port\,2} \tag{B.4}$$

B.3.2 Return Loss

Return Loss (RL) is an attenuation of a RF signal caused by impedance mismatch between a source and a load. Although the impedance matching circuit is calculated and designed correctly, impedance components will always be introduced by component imperfections and surrounding effects like copper traces nearby on the PCB. The return loss is often measured as the ratio between the signal inserted and the signal reflected from the load. Return loss is given by Eq. B.5:

$$RL\,[dB] = 10\log\left(S_{21}\right) \tag{B.5}$$

B.3.3 Insertion Loss

Insertion Loss (IL) is a measure of the attenuation caused by the conductors resistance and the radiated energy in a load. IL is independent of impedance matching between source and load.
IL is given by the equation:

$$IL\,[dB] = 10\log\left(S_{12}\right) \tag{B.6}$$

Appendix C

Antenna Theory

This appendix contains an introduction into various topics within antenna theory and explains terms and concepts discussed in chapter 5.

An antenna is a transducer which transforms a high frequency electrical signal into an electromagnetic wave and vice versa. The device is crucial in a radio system, it acts as an interface between the electric and electromagnetic domain. When an antenna receives an electromagnetic signal, the wave will induce an electrical voltage which will vary in time with the same frequency as the wave. When transmitting the antenna will radiate an electromagnetic wave with the same frequency as the signal delivered from the transmission lines.

C.1 Electromagnetic Waves

An electromagnetic wave is a phenomenon where a electric field and a magnetic field is oscillating perpendicular to each other and perpendicular to the direction both of them are transferring. This phenomenon is described by the Maxwell equations which explain how an electric field generates a magnetic field and a magnetic field generates an electric field, thus creating a self-sustaining electromagnetic wave. The electromagnetic wave can be explained both through a wave model and a particle model. In radio communication, for obvious reasons the wave model is used. The electromagnetic spectrum is divided into several types. The types are classified according to the wavelengths (λ). The whole range covers radio waves, micro waves, infra red light, visible light, ultra violet light and gamma rays.

$$\lambda = \frac{c}{f} \tag{C.1}$$

Radio signals are located in the lower part of the electromagnetic spectrum and has a frequency that is lower than infra red (IR) light. Radio signals

cover the traditional term of radio waves which extends up to 1GHz as well as microwaves because many radio communication systems today use frequencies well above 1GHz.

C.1.1 Maxwell's Equations

Maxwell's equations are a mathematical description of the relationship between electric and magnetic fields. The equations were developed by James C. Maxwell and he predicted the existence of electromagnetic waves [24]. The equations were later used by H.R. Hertz to create electromagnetic waves for the first time in 1887.
There are four equations:

$$\oint \overrightarrow{E} \bullet d\overrightarrow{A} = \frac{Q_{encl}}{\epsilon_0} \qquad \left(Gauss's\, law\, of\, electricity\right) \tag{C.2}$$

$$\oint \overrightarrow{B} \bullet d\overrightarrow{A} = 0 \qquad \left(Gauss's\, law\, of\, magnetism\right) \tag{C.3}$$

$$\oint \overrightarrow{B} \bullet d\overrightarrow{l} = \mu_0 \left(i_c + \epsilon_0 \frac{d\Phi_E}{dt} \right)_{encl} \qquad \left(Ampere's\, law\right) \tag{C.4}$$

$$\oint \overrightarrow{B} \bullet d\overrightarrow{l} = -\frac{d\Phi_B}{dt} \qquad \left(Ampere's\, law\right) \tag{C.5}$$

C.1.2 Polarization

Electromagnetic waves can have different polarizations. Polarization is defined by the movement of the wave in two dimensions (x and y-axis) while propagating in a third dimension (z-axis). The direction the wave is propagating in is called the Poynting vector. A wave can have three types of polarization, linear, circular or elliptical.

Linear

When the electrical field has the same phase as the magnetic field, the electromagnetic wave will only move in a straight line seen from the Poynting vector. This is called a linear polarization and can be both horizontal and vertical. Horizontal polarization is when the electric field moves along the horizontal axis of the of the Poynting vector. Vertical polarization is when the electric field moves along the vertical axis of the Poynting vector.

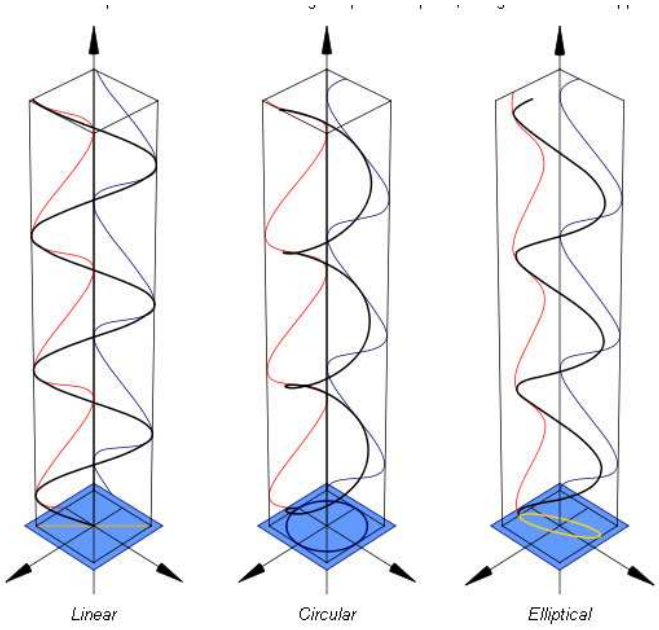

Figure C.1: Linear, circular and elliptical polarized waves. Credited: Wikipedia

Circular

If the electric field has exactly 90^o phase difference from the magnetic field the wave will move in a circular movement around the Poynting vector. There are two types of circular polarization. Right-hand circular polarization (RHCP) and left-hand circular polarization (LHCP). The two are defined by the direction of rotation around the Poynting vector seen towards the receiver.

Elliptical

Elliptical polarization is defined as an electric field with a phase difference other than 90^o and/or different amplitude than the magnetic field. The wave will rotate, but will vary in amplitude over time. An elliptical polarized wave is characterized by the ratio between maximum and minimum values of the electric field, the so-called axial ratio (AR).

$$AR = \frac{E_{max}}{E_{min}} \tag{C.6}$$

C.1.3 Polarization Mismatch

Polarization mismatch is a term describing the attenuation caused by a receiving antenna and an incoming signal that has different polarization.

C.2 Isotropic Antenna

An Omni-directional antenna (e.g. isotropic antenna) is an antenna that transmits and receives a wave with equal gain in all directions. This is a theoretical concept used as a reference to compare the gain of practical antennas. All antennas are measured in $[dBi]$ which refers to an isotropic antenna with $Gain = 1$ or $0dBi$.
Radiated power from an omni-directional source always has a gain equal to 1. In practical designs the radiation pattern of the half-wave dipole antenna is closest in comparison to the isotropic antenna, and is sometimes also used as a reference using the term $[dBd]$.

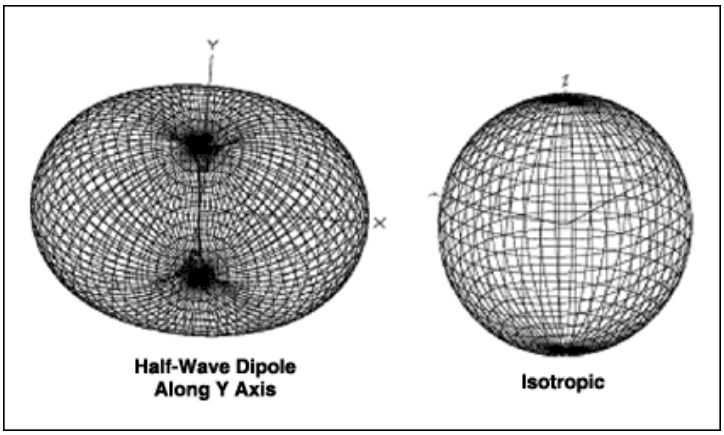

Figure C.2: Radiation pattern of a dipole antenna and an isotropic antenna. Credited: Wikipedia

C.3 Antenna Characteristics

This section will explain some of the key properties describing the characteristics of an antenna.

C.3.1 Radiation Pattern

The radiation pattern is a graphic representation of how the energy transmitted from an antenna is distributed in a 3D space. The plot indicates the energy level per unit angle. The plot can be both a 3 dimensional plot as

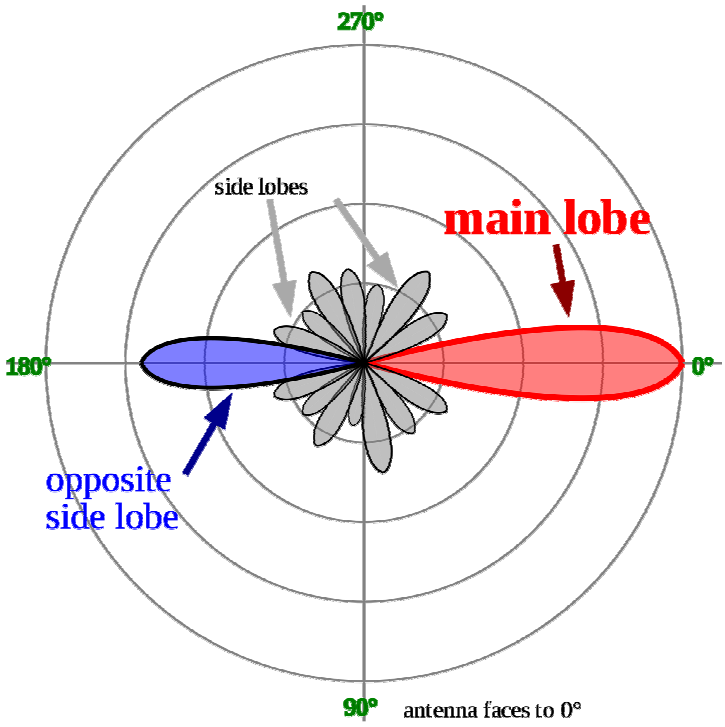

Figure C.3: 2D plot of the radiation pattern of a generic antenna. Credited: Wikipedia

seen in figure C.2 or a 2 dimensional as seen in C.3. From the plot in figure C.3 some of the key parameters can be found:

- The *half-power beamwidth*, normally just called the beamwidth is the angle on the main lobe from the peak power to where the power is reduced by 3dB
- The *front-back ratio* is the ratio between the peak power on the front lobe and the back lobe
- The *side lobe level* is the peak power of the biggest side lobe

C.3.2 Directivity

Directivity is a ratio of the radiation pattern of an antenna compared to the radiation pattern of an isotropic antenna transmitting with the same power.

$$D(\theta,\phi)[dBi] = \frac{Radiation\,intensity\,of\,an\,antenna\,in\,direction\,(\theta,\phi)}{Radiation\,intensity\,of\,an\,isotropic\,antenna} \tag{C.7}$$

C.3.3 Bandwidth

The bandwidth of an antenna is defined as the frequency band in which $P_{out} \geq P_{max}/2$. The parameters of an antenna only apply for frequencies within the specified bandwidth.

C.3.4 Efficiency

The *efficiency* of an antenna is defined as the ratio between the applied power and the transmitted power.

$$efficiency(e) = \frac{P_{radiated}}{P_{applied}} \tag{C.8}$$

C.3.5 Power Gain

The power gain is the actual gain of an antenna. It is defined as the ratio between the power radiated and an isotropic antenna. In theoretical applications where the antennas are assumed to be ideal, the gain would be the same as D_{max}, but in practical applications the antenna efficiency influences the gain.

$$Gain[dBi] = eD(\theta, \phi)[dBi] \tag{C.9}$$

In most cases, the antenna datasheet specifies the gain of an antenna, this is the ratio between the peak power in the main lobe and an isotropic antenna.

$$Gain[dBi] = G_{max}[dBi] = eD(0^o, 0^o)[dBi] = eD_{max}[dBi] = \tag{C.10}$$

C.4 Antenna Types

There are many types of antennas, some are designed for high directional gain like Yagi antennas, others for high frequencies like parabolic dishes and others for a wide bandwidth. The Cubesat satellites have little or no attitude control while in orbit. Because of this it is common to us antennas with near omni-directional gain. This section will describe the antenna types commonly used by Cubesat satellites.

C.4.1 Monopole Antenna

The monopole antenna also known as a *quarter-wave antenna*, is in fact a type of dipole antenna where one of the elements is replaced with a ground plane. A monopole antenna is the simplest form of antennas, commonly used in mobil phone applications, cars and etc.
The monopole antenna is considered nearly omni-directional which means that it has no gain. The name quarter-wave antenna comes from the physical dimension where the antenna element is approximately 1/4 of the wave

length of the operating frequency. A monopole antenna can only transmit linear polarized waves (se C.1.2).

C.4.2 Dipole Antenna

A dipole antenna is an antenna built by two quarter-wave elements placed close to each other to achieve a physical length of half the wavelength of the operating frequency. The signal elements are terminated into a signal source where one is connected to the signal line and one to the ground plane.
like the monopole antenna the dipole has a radiation pattern close to omni-directional which can be seen in figure C.2 and only transmit linear polarized radio waves.

C.4.3 Turnstile

A turnstile antenna, also known as a *crossed dipole* antenna, is a an antenna with two dipole antennas in a crossed configuration (see figure C.4). The

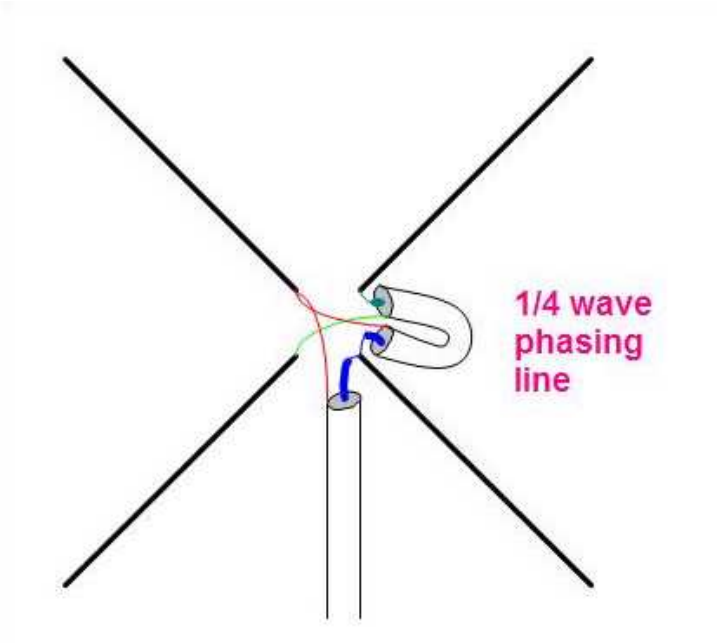

Figure C.4: A turnstile antenna with a 90^o phase shift line on one of the dipole antennas.

signal to one of the dipoles are phase shifted with 90^o making the turnstile transmit a circular polarized signal. The antenna is omni-directional

Appendix D

Miscellanous Work

D.1 Presentations

- Presentation at the 6th Annual Cubesat Developers Workshop, California Polytechnic State University, San Luis Obispo, USA, 22-25 Apr. 2009
- Propulsion System Conference, NASA Ames Research Center, California, USA, 8. Aug. 2009
- Presentation at the ANSAT Cubesat Workshop, Andoya Rocket Range, 23-24 Nov. 2009
- Presentation at the ANSAT Cubesat Workshop, Andoya Rocket Range, 3-4 May 2010

D.2 Technical Documents

- Technical document, CubeSTAR Backplane Signal Defintion
- Technical document, CubeSTAR I2C Bus Memory Map
- Work document, Radiation Tolerance Testing of COTS in the CubeSTAR Satellite

D.3 Activities

- 6th Annual Cubesat Developers Workshop, California Polytechnic State University, San Luis Obispo, 22-25 Apr. 2009
- International Space University, Space Studies Program 2009
  hosted by NASA Ames Research Center, California, USA

- Cubesat Workshop, Andoya Rocket Range, 23-24 Nov. 2009
- Cubesat Workshop, Andoya Rocket Range, 3-4 May 2010

Appendix E

Link Budget

This appendix contains the link budget for the uplink and downlink as well as a summary of the link budget. The appendix is linked to chapter 3.
The link budget has been produced using the AMSAT / IARU Annotated Link Model System ver.2.4.1.

CubeSTAR Uplink Command Budget:	NOTE:		CubeSTAR Version: 2.4.1	Date Data Last Modified: 2009 September 23
Parameter:	*Value:*	*Units:*	*Comments:*	
Ground Station:				
Ground Station Transmitter Power Output:	75,0	watts	This value is transferred from "Transmitters" W/S, Cell [E15].	
In dBW:	18,8	dBW	Transmitter power expressed in dB above one watt	
In dBm:	48,8	dBm	Transmitter power expressed in dB above one milliwatt	
Ground Stn. Total Transmission Line Losses:	7,3	dB	This value is transferred from "Transmitters" W/S, Cell [I33]	
Antenna Gain:	22,0	dBi	This value is selected at "Antenna Gain" W/S, Cell [E11]	
Ground Station EIRP:	33,4	dBW	Ground Station Effective Isotropic Radiated Power (EIRP) [EIRP=Pt x Lti x Ga]	
Uplink Path:				
Ground Station Antenna Pointing Loss:	0,3	dB	This value is calculated in the "Antenna Pointing Losses" W/S, and transferred from Cell [K43]	
Gnd-to-S/C Antenna Polarization Losses:	0,2	dB	This value is calculated in the "Polarization Loss" W/S and is transferred from Cell [F40].	
Path Loss:	151,0	dB	Lp = 22 + 20LOG(D/λ); Transferred from "Orbit & Frequency" W/S	
Atmospheric Losses:	1,1	dB	This value is transferred from "Atmos. & Ionos. Losses" W/S, Cell [D23]	
Ionospheric Losses:	0,4	dB	This value is transferred from "Atmos. & Ionos. Losses" W/S, Cell [D47:D50]	
Rain Losses:	0,0	dB	This value should be estimated by the link model operator and place into Cell [B16]	
Isotropic Signal Level at Spacecraft:	-119,6	dBW	This is the signal level received in space in the vacinity of the spacecraft using an omnidirectional antenna.	
Spacecraft (Eb/No Method):				
------ Eb/No Method ------				
Spacecraft Antenna Pointing Loss:	0,1	dB	This value is transferred from "Antenna Pointing Losses" W/S, Cell [K63]	
Spacecraft Antenna Gain:	2,0	dBi	This value is selected at "Antenna Gain" W/S, Cell [E24]	
Spacecraft Total Transmission Line Losses:	0,5	dB	This value is transferred from the "Receivers" W/S, Cell [J52]	
Spacecraft Effective Noise Temperature:	428	K	This value is calculated in the "Receivers" W/S and Transferred from Cell [J67]	
Spacecraft Figure of Merrit (G/T):	-24,8	dB/K	G/T = Ga-Lsi-10log(Ts). This is the ultimate measure of the receiver's performance.	
S/C Signal-to-Noise Power Density (S/No):	84,1	dBHz	Boltzmans Constant: -228,6 dBW/K/Hz	
System Desired Data Rate:	9600	bps	Operator selects this value. Be Careful! This is the data rate, not the symbol rate.	
In dBHz:	39,8	dBHz	This is simply = 10log(R); R= data rate	
Command System Eb/No:	44,3	dB		
Demodulation Method Seleted:	Non-Coherent FSK		Values selected in "Modulation-Demodulation W/S, Cell [E3]	
Forward Error Correction Coding Used:	None		Value selected in "Modulation-Demodulation" W/S, also Cell [E3]	
System Allowed or Specified Bit-Error-Rate:	1,0E-05		The selected value is transferred from the "Modulation-Demodulation W/S, Cells [E6:E23]	
Demodulator Implementation Loss:	1,0	dB	This value is transferred from the "Modulation-Demodulation W/S, Cell[E25]	
Telemetry System Required Eb/No:	13,8	dB	The selected value is transferred from the "Modulation-Demodulation W/S, Cells [F6:F23]	
Eb/No Threshold:	14,8	dB	This is the result of the "Modulation-Demodulation" W/S and is transferred from Cell [H32]	
System Link Margin:	29,5	dB		
Spacecraft Alternative Signal Analysis Method (SNR Computation):			← NOTE:	
-------- SNR Method ---------				
Spacecraft Antenna Pointing Loss:	0,1	dB	This value is transferred from "Antenna Pointing Losses" W/S, Cell [K63]	
Spacecraft Antenna Gain:	2,0	dBi	This value is selected at "Antenna Gain" W/S, Cell [E24]	
Spacecraft Total Transmission Line Losses:	0,5	dB	This value is transferred from the "Receivers" W/S, Cell [J52]	
Spacecraft Effective Noise Temperature:	428	K	This value is calculated in the "Receivers" W/S and Transferred from Cell [J67]	
Spacecraft Figure of Merrit (G/T):	-24,8	dB/K	G/T = Ga-Lsi-10log(Ts). This is the ultimate measure of the receiver's performance.	
Signal Power at Spacecraft LNA Input:	-118,2	dBW	Ps = Piso+Ga-Lpi-Lti; This is the signal power that has arrived at the ground station receiver.	
Spacecraft Receiver Bandwidth:	30 000	Hz	Signal Spectrum Must Pass Through This Data Filter. NOTE:	
Spacecraft Receiver Noise Power (Pn = kTB)	-157,5	dBW	Pn = K + 10log(Ts) + 10log(B). This is the total noise power arriving at the ground station receiver.	
Signal-to-Noise Power Ratio at G.S. Rcvr:	39,3	dB	Ps/Pn = Ps(in dBW) - Pn(in dBW)	
Analog or Digital System Required S/N:	13,8	dB	If system is digital, use values from "Modulation-Demodulation" W/S. If analog, use appropriate value from text book.	
System Link Margin	25,5	dB		

Figure E.1: CubeSTAR Uplink Budget

CubeSTAR	NOTE:	CubeSTAR	Date Data Last Modified:
Downlink Telemetry Budget:		Version: 2.4.1	2009 September 23

Parameter:	Value:	Units:	Comments:
Spacecraft:			
Spacecraft Transmitter Power Output:	1,0	watts	This value is transferred from "Transmitters" W/S, Cell [E50]
In dBW:	0,0	dBW	Transmitter power expressed in dB above one watt
In dBm:	30,0	dBm	Transmitter power expressed in dB above one milliwatt
Spacecraft Total Transmission Line Losses:	0,6	dB	This value is transferred from "Transmitters" W/S, Cell [I68]
Spacecraft Antenna Gain:	2,2	dBi	This value is selected at "Antenna Gain" W/S, Cell [E41]
Spacecraft EIRP:	1,6	dBW	Spacecraft Effective Isotropic Radiated Power (EIRP) [EIRP=Pt x Ltl x Ga]
Downlink Path:			
Spacecraft Antenna Pointing Loss:	0,1	dB	This value is calculated in the "Antenna Pointing Losses" W/S, and trasferred from Cell [K85]
S/C-to-Ground Antenna Polarization Loss:	0,2	dB	This value is calculated in the "Polarization Loss" W/S and is transferred from Cell [F60].
Path Loss:	151,0	dB	Lp = 22 + 20LOG(D/λ): Transferred from "Orbit & Frequency" W/S
Atmospheric Loss:	1,1	dB	This value is transferred from "Atmos. & Ionos. Losses" W/S, Cell [D23]
Ionospheric Loss:	0,4	dB	This value is transferred from "Atmos. & Ionos. Losses" W/S, Cell [D47:D50]
Rain Loss:	0,0	dB	This value should be estimated by the link model operator and place into Cell [B18]
Isotropic Signal Level at Ground Station:	-151,3	dBW	This is the signal level received at the Earth in the vacinity of the ground station using an omnidirectional antenna.
Ground Station (EbNo Method):			
----- Eb/No Method -----			
Ground Station Antenna Pointing Loss:	0,3	dB	This value is transferred from "Antenna Pointing Losses" W/S, Cell [K102]
Ground Station Antenna Gain:	22,0	dBi	This value is selected at "Antenna Gain" W/S, Cell [E58]
Ground Station Total Transmission Line Losses:	2,7	dB	This value is transferred from the "Receivers" W/S, Cell [J123]
Ground Station Effective Noise Temperature:	352	K	This value is calculated in the "Receivers" W/S and Transferred from Cell [J138]
Ground Station Figure of Merrit (G/T):	-6,1	dB/K	G/T = Ga-Lr-10log(Ts). This is the uplimate measure of the receiver's performance.
G.S. Signal-to-Noise Power Density (S/No):	70,9	dB-Hz	Boltzmen's Constant: -228,6 dBW/K/Hz
System Desired Data Rate:	9600	bps	Operator selects this value. Be Careful! This is the data rate, not the symbol rate.
In dBHz:	39,8	dBHz	This is simply = 10log(R); R= data rate
Telemetry System Eb/No for the Downlink:	31,0	dB	
Demodulation Method Seleted:	Non-Coherent PSK		Values selected in "Modulation-Demodulation W/S, Cell [E30]
Forward Error Correction Coding Used:	None		Value selected in "Modulation-Demodulation" W/S, also Cell [E30]
System Allowed or Specified Bit-Error-Rate:	1,0E-05		The selected value is transferred from the "Modulation-Demodulation W/S, Cells [E33:E50]
Demodulator Implementation Loss:	0	dB	This value is transferred from the "Modulation-Demodulation W/S, Cell [E52]
Telemetry System Required Eb/No:	13,8	dB	The selected value is transferred from the "Modulation-Demodulation W/S, Cells [F33:F50]
Eb/No Threshold:	13,8	dB	This is the result of the "Modulation-Demodulation" W/S and is transferred from Cell [H32]
System Link Margin:	17,2	dB	
Ground Station Alternative Signal Analysis Method (SNR Computation):			
---------- SNR Method ----------			
Ground Station Antenna Pointing Loss:	0,3	dB	This value is transferred from "Antenna Pointing Losses" W/S, Cell [K102]
Ground Station Antenna Gain:	22,0	dBi	This value is selected at "Antenna Gain" W/S, Cell [E58]
Ground Station Total Transmission Line Losses:	2,7	dB	This value is transferred from the "Receivers" W/S, Cell [J123]
Ground Station Effective Noise Temperature:	352	K	This value is calculated in the "Receivers" W/S and Transferred from Cell [J138]
Ground Station Figure of Merrit (G/T):	-6,1	dB/K	G/T = Ga-Lr-10log(Ts). This is the ultimate measure of the receiver's performance.
Signal Power at Ground Station LNA Input:	-132,3	dBW	Ps = Pt-o-t-Ga-Lp-Lr-Lt; This is the signal power that has arrived at the ground station receiver.
Ground Station Receiver Bandwidth (B):	30.000	Hz	Signal Spectrum Must Pass Through This Data Filter NOTE:
G.S. Receiver Noise Power (Pn = kTB)	-158,4	dBW	Pn = K + 10log(Ts) + 10log(B). This is the total noise power arriving at the ground station receiver.
Signal-to-Noise Power Ratio at G.S. Rcvr:	26,1	dB	Ps/Pn = Ps(in dBW) - Pn(in dBW)
Analog or Digital System Required S/N:	13,8	dB	If system is digital, use values from "Modulation-Demodulation" W/S. If analog, use appropriate value from text book.
System Link Margin	12,3	dB	

Figure E.2: CubeSTAR Downlink Budget

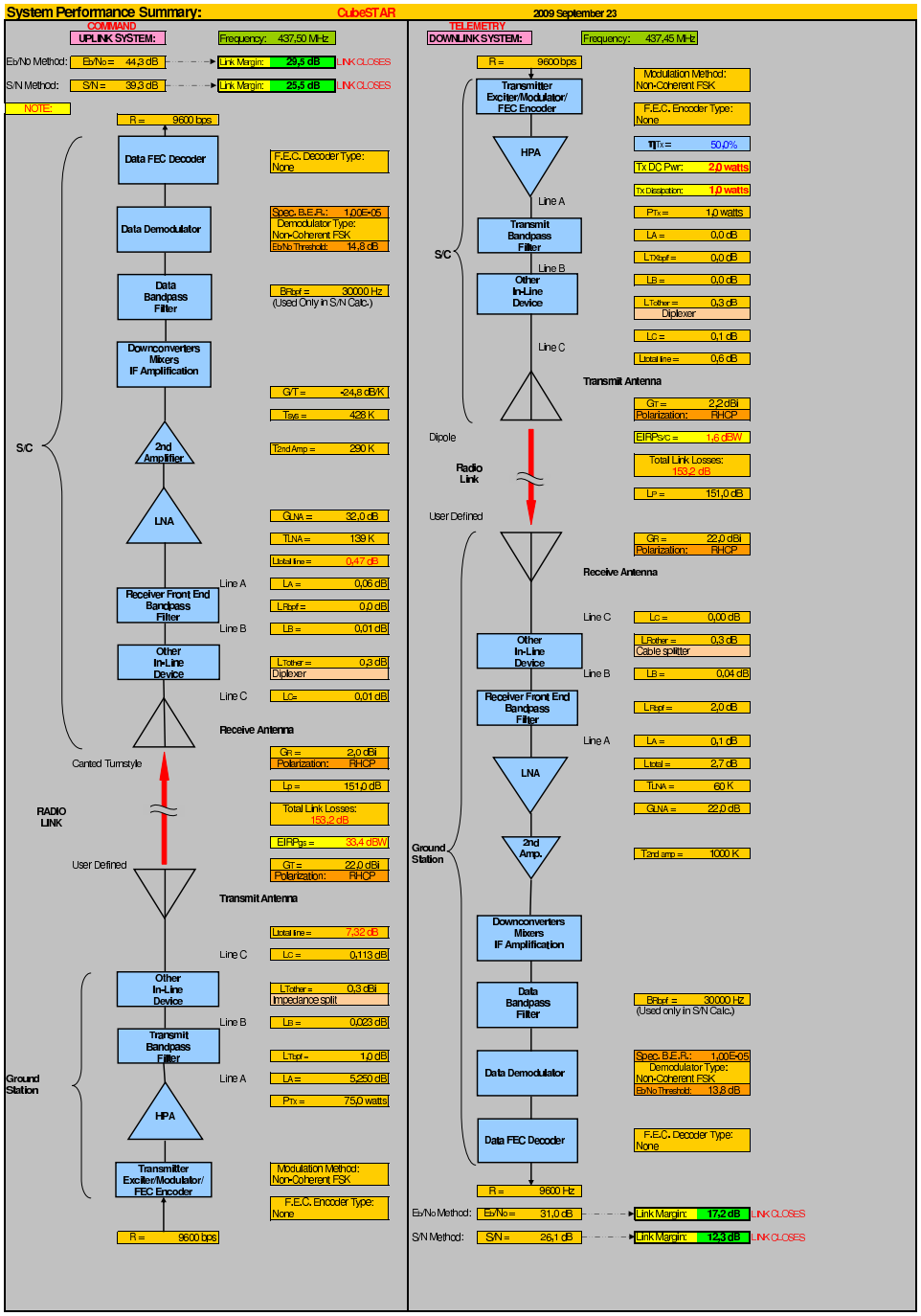

Figure E.3: CubeSTAR Link Budget Summary

Appendix F

Schematic

This appendix contains the parts list, schematics and PCB from the prototype system. The parts list, schematic and PCB has been produced using the CADSTAR packet v.12 from Zuken.

Listing F.1: Parts list for the communication system

```
                                Parts List

                        CADSTAR Design Editor Version 12.0.0.1

Design:      CUBESTAR - Communication System

Design Title:

Date:        1. April 2010
```

Part Name	Description	Qty.	Comps.
ATMEL/ATXMEGA128/TQFP	ATMEL AVR MICROCONTROLLER	1	U1
CAP/100NF/0603R	10% 16V 0603 X7R	3	C1
			C5-6
CAP/100PF/0402R	10% 50V 0402 NP0 7"REEL	4	C22-24
			C53
CAP/12PF/0402R	5% 50V 0402 NP0 7"REEL	2	C14-15
CAP/15PF/0603R	E5% 50V 0603 NP0	2	C54-55
CAP/220NF/1206R	20% 50V 1206 X7R	2	C2
			C4
CAP/BYPASS/0603R	10% 16V 0603 X7R	4	CB2
			CB4
			CB6
			CB8
CON/AVRJTAG	10PIN FLAT CABLE CON. T&B	1	CN2
CON/PR10	10 SCOTT ELEC. PINROW	1	CN8
CON/PR13X2PIN/HORIZ	13X2 TYCO PINROW ANGELED	1	CN1
CON/PR2	2 SCOTT ELEC. PINROW	1	CN3
CON/R124_426_123/	SMA-CONNECTOR, PCB	1	CN4
EMC_COVER_LONG	SCREEN WALL LONG SIDE	2	X5-6
EMC_COVER_SHORT	SCREEN WALL SHORT SIDE	2	X8
			X10
IND/1U0H/NR3010T1R0N	+/-30% 1U0H POWER CHOKE	1	L7
LED/19-21SDRC/SMD	SMD LED RED	2	D1-2
LED/19-21SYGC/SMD	SMD LED GREEN	3	D6-8
MURATA/100PF/0402R	+/-5% 50V 0402 NP0 GRM1555C	1	C43
MURATA/10NF/0402R	+/5% 50V 0402 NP0 GRM1555C	7	C21
			C27
			C30
			C33-34
			C47-48
MURATA/10PF/0402R	+/-5% 50V 0402 NP0 GRM1555C	3	C41
			C46
			C49
MURATA/15PF/0402R	+/-5% 50V 0402 NP0 GRM1555C	1	C42

	MURATA/18NH/0402R	+/−5% 0402 LQG15H−series	3	L10−12
	MURATA/1N0F/0402R	+/5% 50V 0402 NP0 GRM1555C	10	C9
				C28
54				C31
				C40
				C52
				C58−62
	MURATA/220PF/0402R	+/5% 50V 0402 NP0 GRM1555C	5	C13
59				C17
				C20
				C25−26
	MURATA/22NH/0402R	+/−5% 0402 LQG15H−series	1	L3
	MURATA/22PF/0402R	+/−5% 50V 0402 NP0 GRM1555C	1	C36
64	MURATA/27NH/0402R	+/−5% 0402 LQG15H−series	6	L1−2
				L4
				L6
				L8−9
	MURATA/2P0F/0402R	+/−0.25PF 50V 0402 NP0 GRM1555C	1	C38
69	MURATA/330PF/0402R	+/5% 50V 0402 NP0 GRM1555C	2	C29
				C32
	MURATA/3P9F/0402R	+/−0.25PF 50V 0402 NP0 GRM1555C	2	C8
				C16
	MURATA/47PF/0402R	+/−5% 50V 0402 NP0 GRM1555C	2	C35
74				C39
	MURATA/56PF/0402R	+/−5% 50V 0402 NP0 GRM1555C	2	C37
				C63
	MURATA/5P6F/0402R	+/−0.5PF 50V 0402 NP0 GRM1555C	1	C19
	MURATA/6N8H/0402R	+/−5% 0402 LQG15H−series	1	L5
79	MURATA/8P2F/0402R	+/−0.5PF 50V 0402 NP0 GRM1555C	1	C18
	POW/TPS2556/SMD	POWER DIST SWITCH, 5A	2	X11−12
	RES/0R00/0603R	RESISTOR KOA 0603 1% 0.1W	2	R3
				R8
	RES/0R00/1206R	RESISTOR KOA 1206 1% 0.25W	1	R47
84	RES/100K/0603R	RESISTOR KOA 0603 1% 0.1W	2	R14
				R16
	RES/10K0/0402R	RES KOA 0402 1% 63mW MINIREEL	4	R9−10
				R12−13
	RES/180R/0402R	RES KOA 0402 1% 63mW MINIREEL	1	R4
89	RES/18R0/0402R	RES KOA 0402 1% 63mW MINIREEL	1	R27
	RES/270K/0603R	RESISTOR KOA 0603 1% 0.1W	1	R17
	RES/47K0/0603R	RESISTOR KOA 0603 1% 0.1W	2	R5
				R7
	RES/56K0/0402R	RES KOA 0402 1% 63mW MINIREEL	1	R25
94	RES/62R0/1206R	RESISTOR KOA 1206 1% 0.25W	5	R1−2
				R11
				R15
				R26
	RES/68K0/0603R	RESISTOR KOA 0603 1% 0.1W	1	R18
99	SPES/CC1101	SUB−1 GHz RF TRANCEIVER	1	X4
	SPES/RF1200	GENERAL PURPOSE/HF SWITSCH	2	X2−3
	SPES/RF3866	GSM LOW NOISE AMPLIFIER	1	X1
	SPES/RF5110G	GSM POWER AMPLIFIER	1	X9
	SW/SKHUAF/SMD	ALPS−SMD PUSH BUTTON	2	SW − RESET
104				SW1 − GENERIC
	TANT/10UF/16V/SMD	TANTAL ELECTROLYTIC CAP	1	C7
	TANT/3U3F/35V/L−SMD	TANTAL ELECTROLYTIC CAP	2	C44−45
	XTAL/16MHZ/SMX	IQD 12SMX SMD XTAL +/− 30PPM	1	XTAL1
	XTAL/C3E−26.000−12−3030−X	CRYSTAL 26MHz SMD	1	X7
109				

End of report

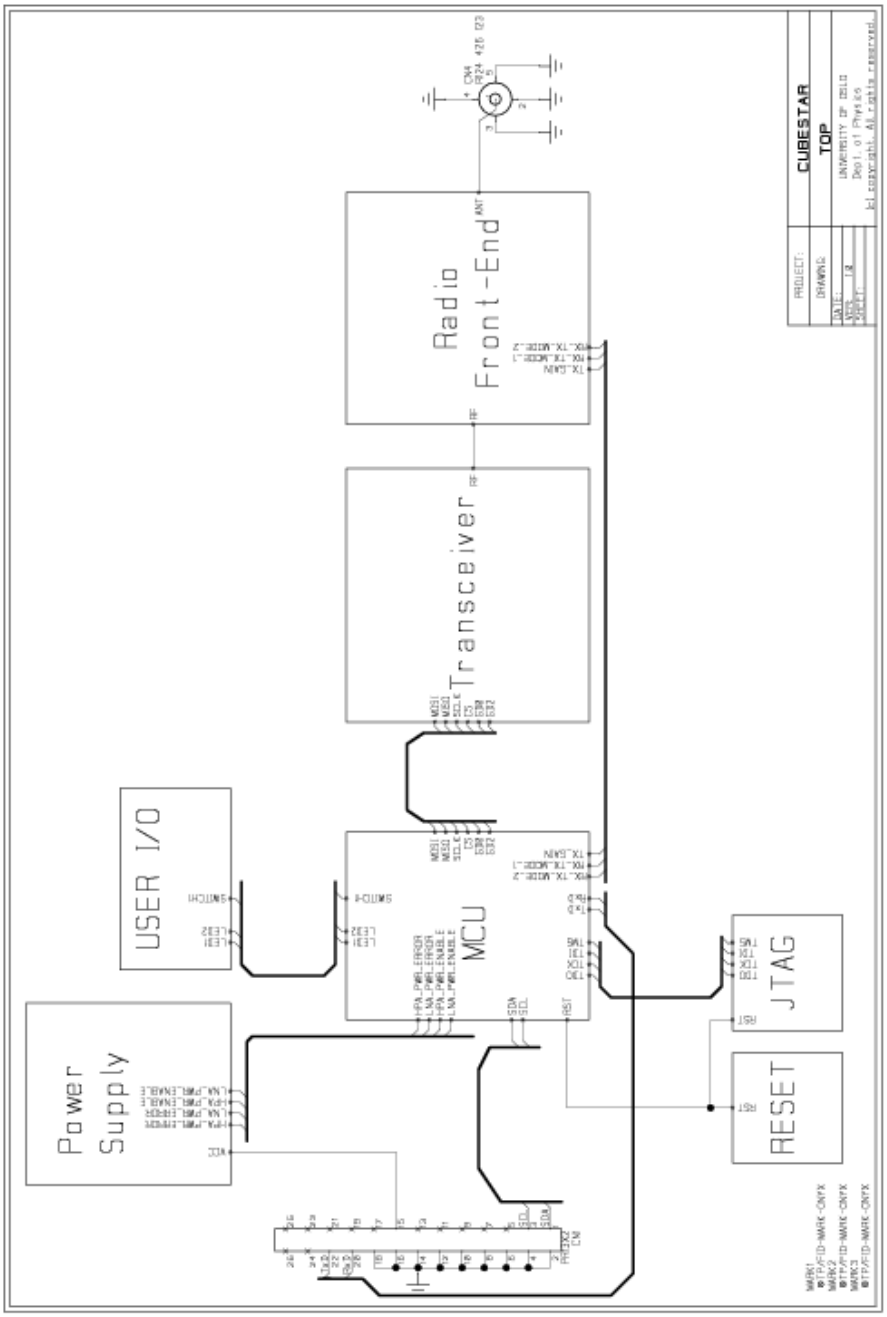

Figure F.1: Schematic top

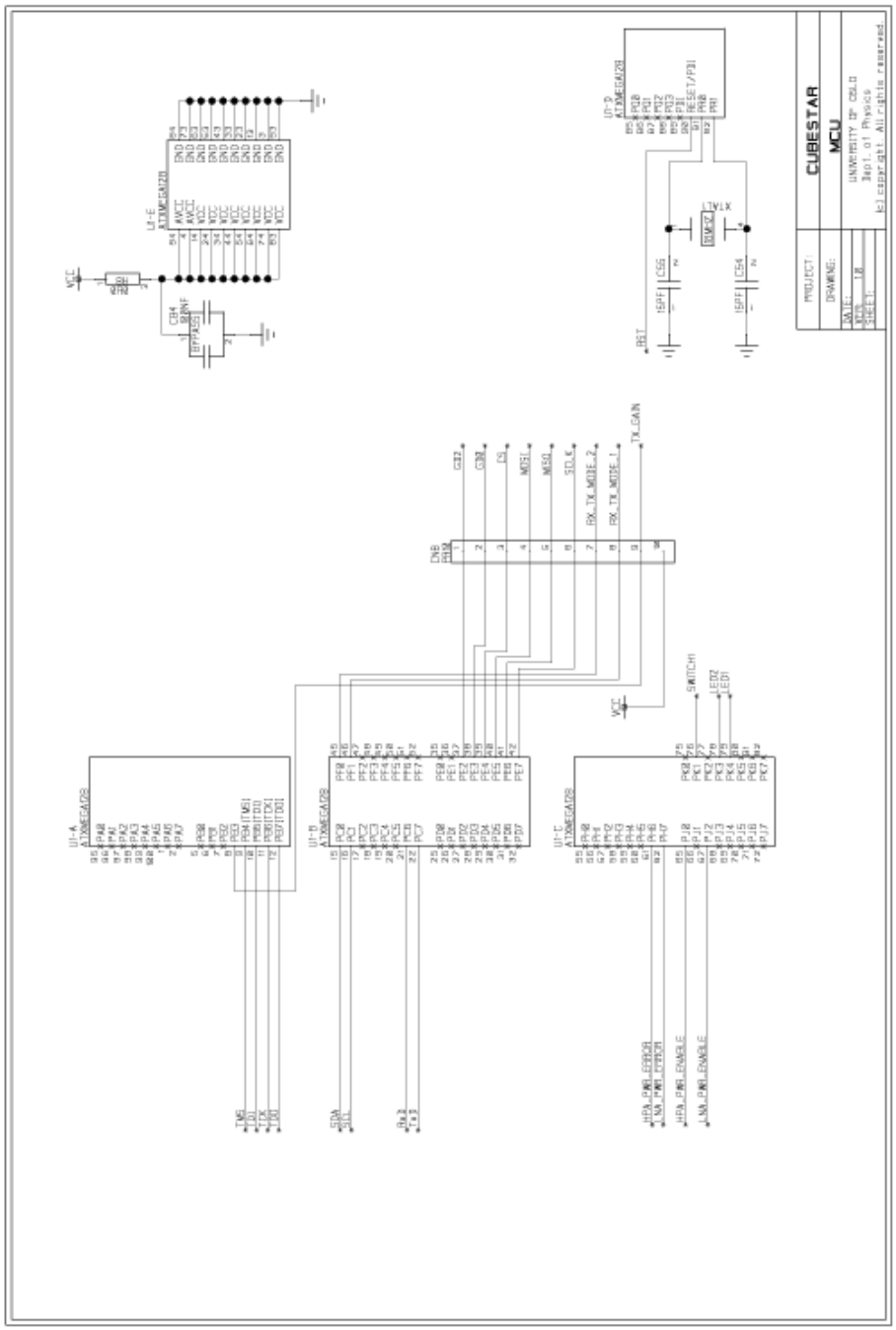

Figure F.2: Schematic MCU

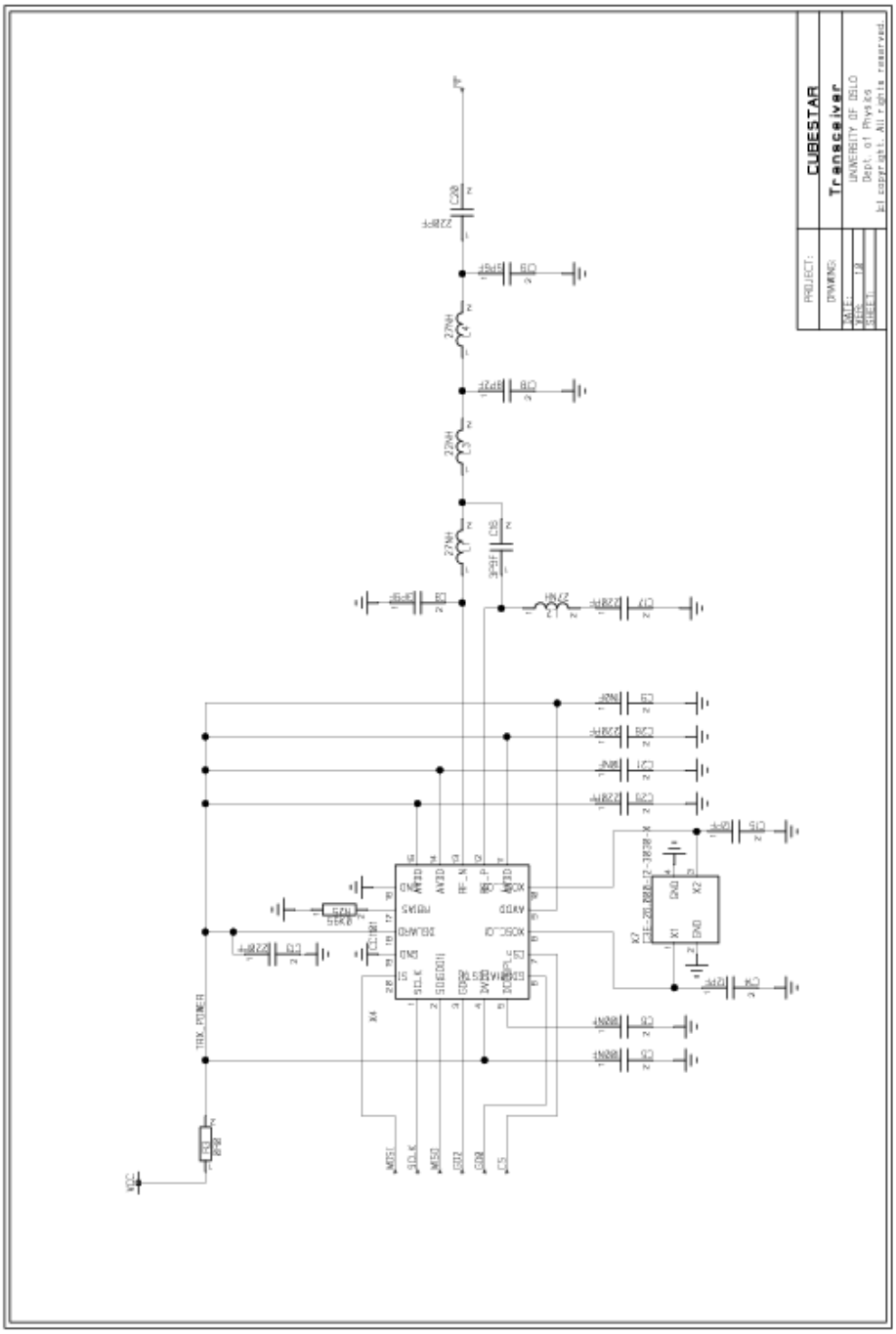

Figure F.3: Schematic transceiver

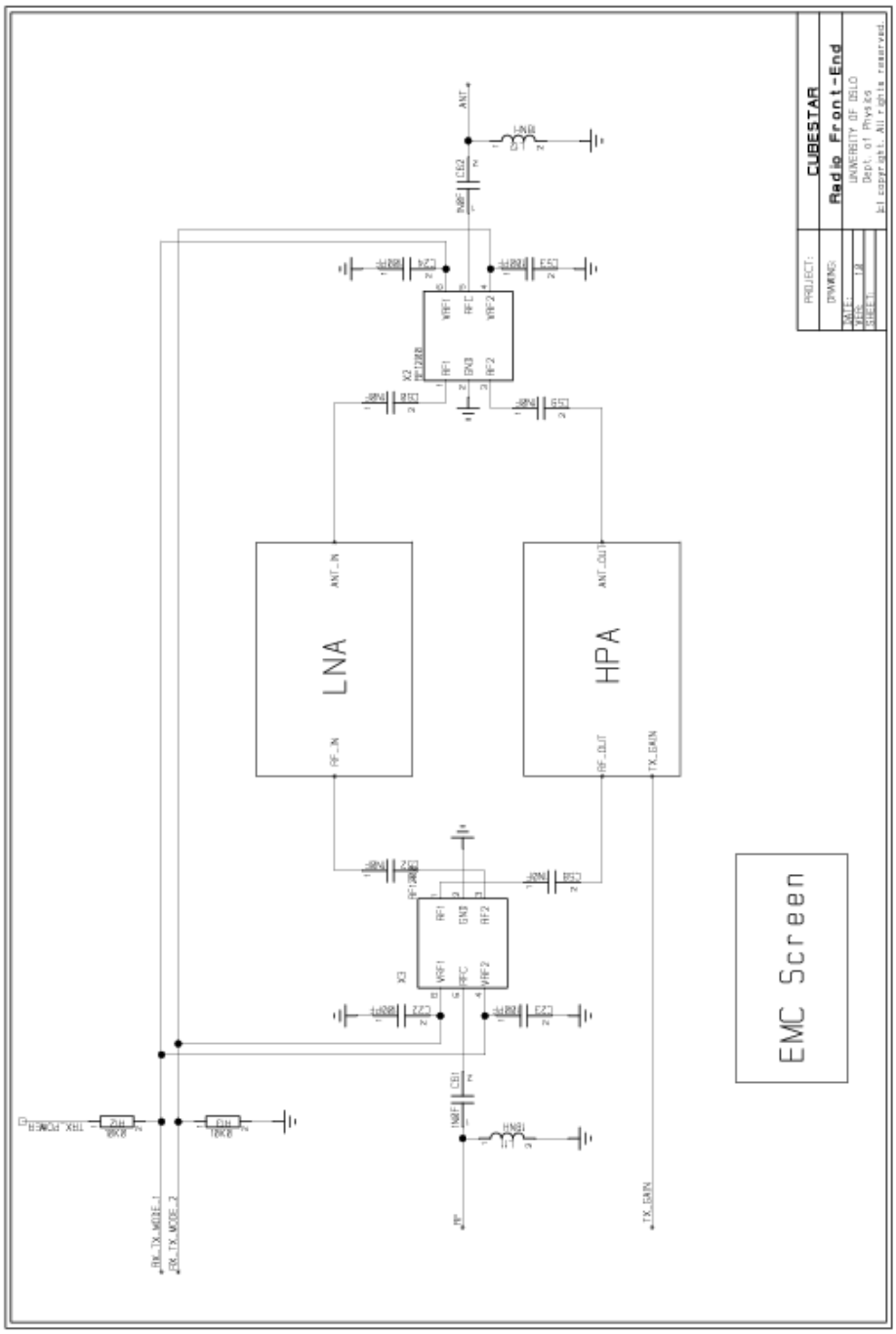

Figure F.4: Schematic Radio Front-End

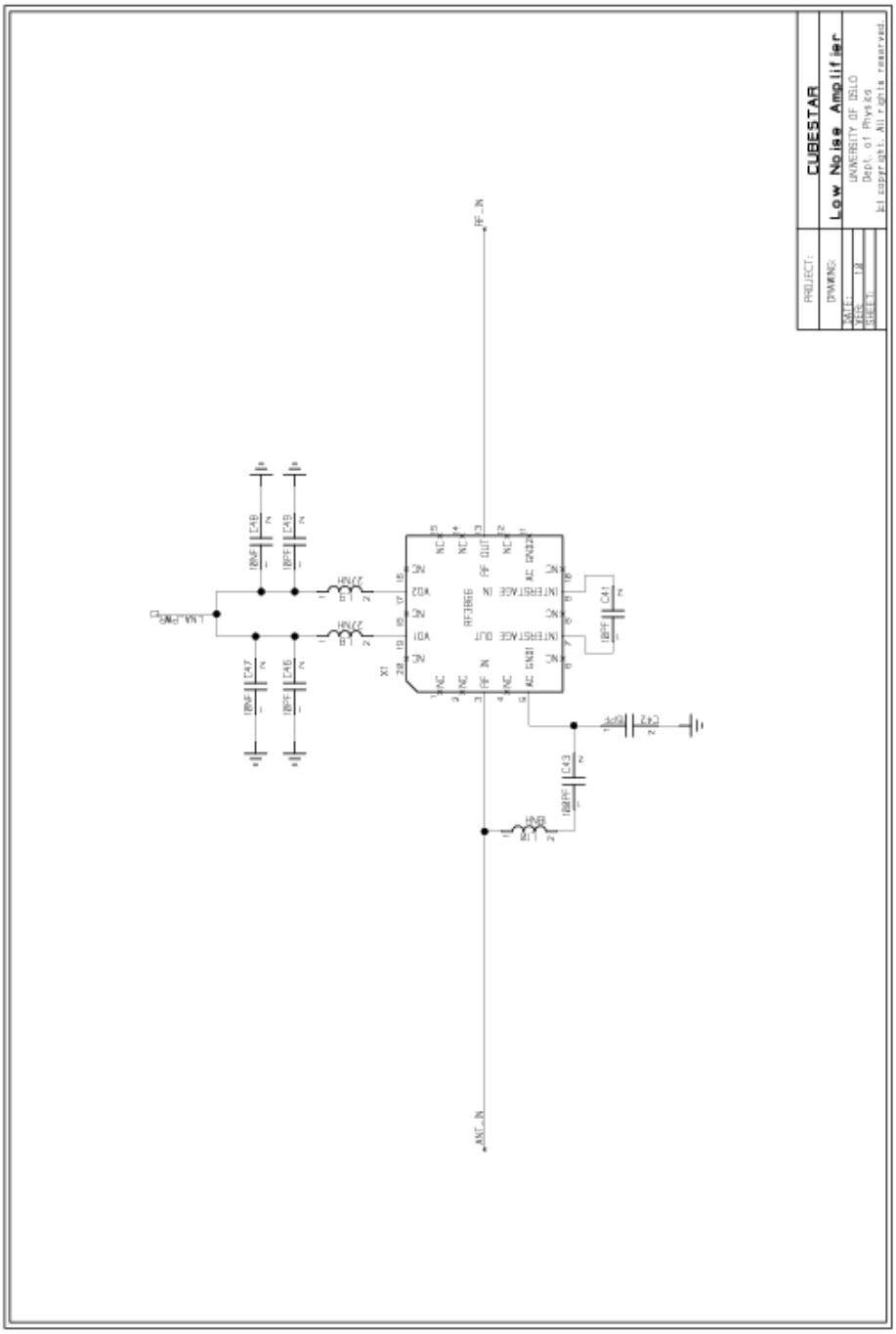

Figure F.5: Schematic Low Noise Amplifier

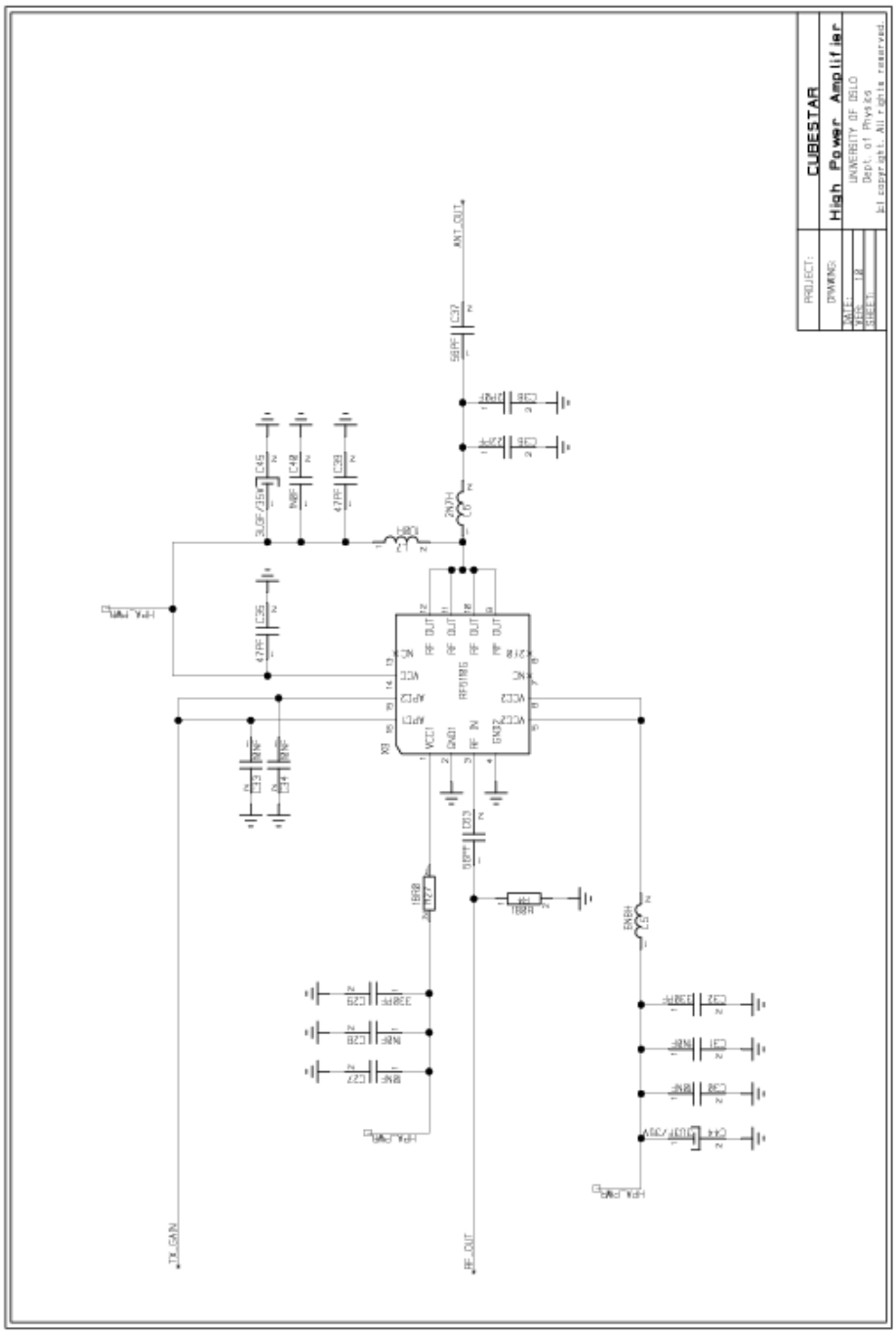

Figure F.6: Schematic High power Amplifier

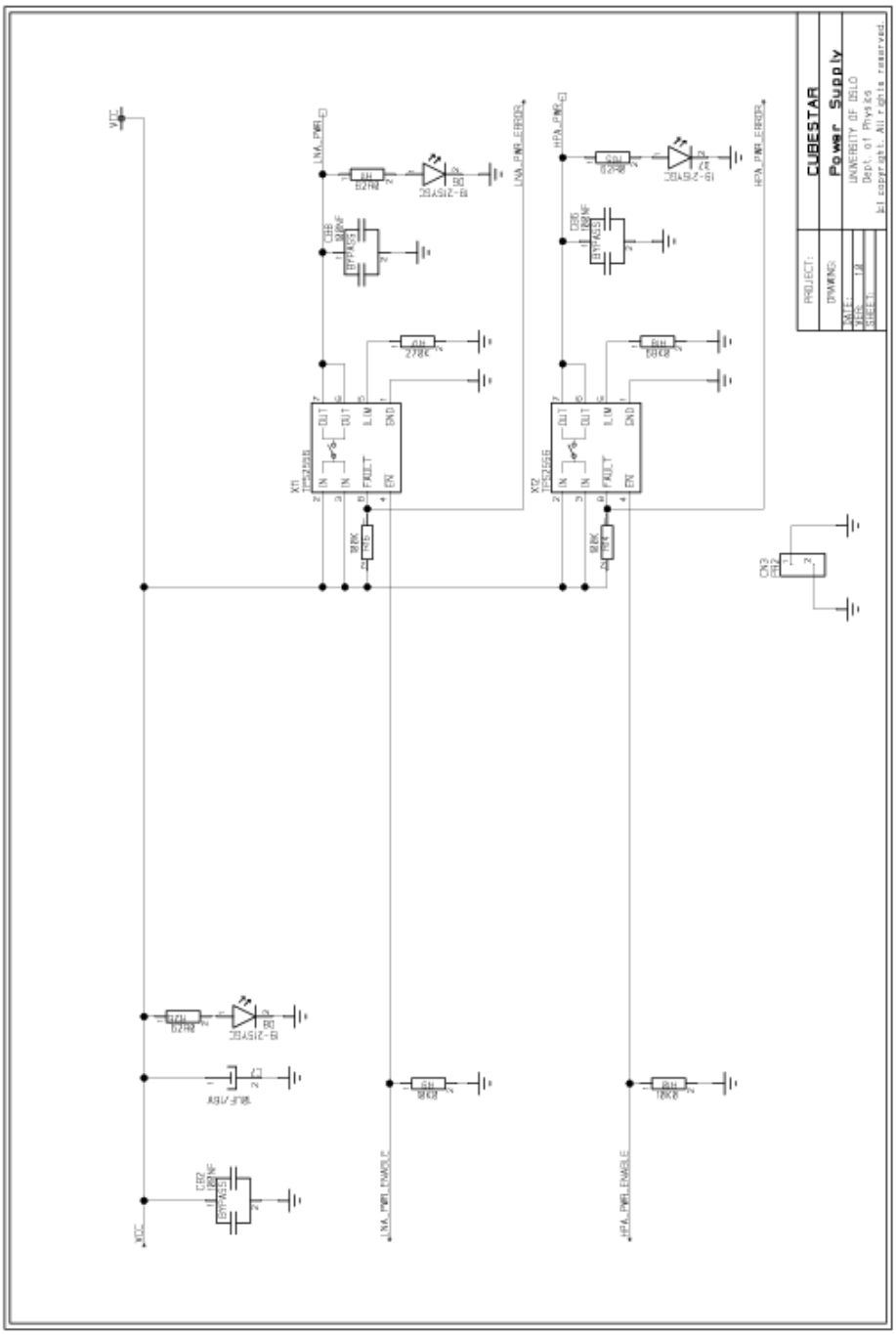

Figure F.7: Schematic Power Control

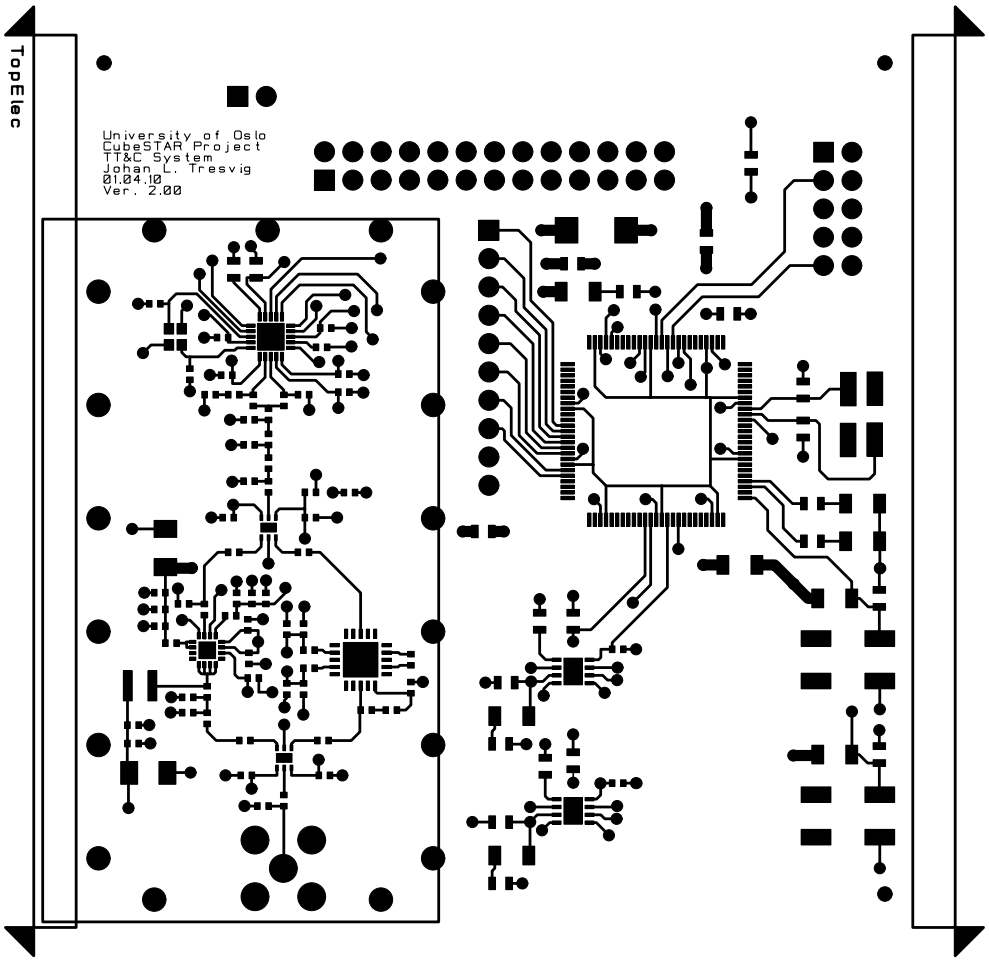

Figure F.8: PCB TopElec Layer

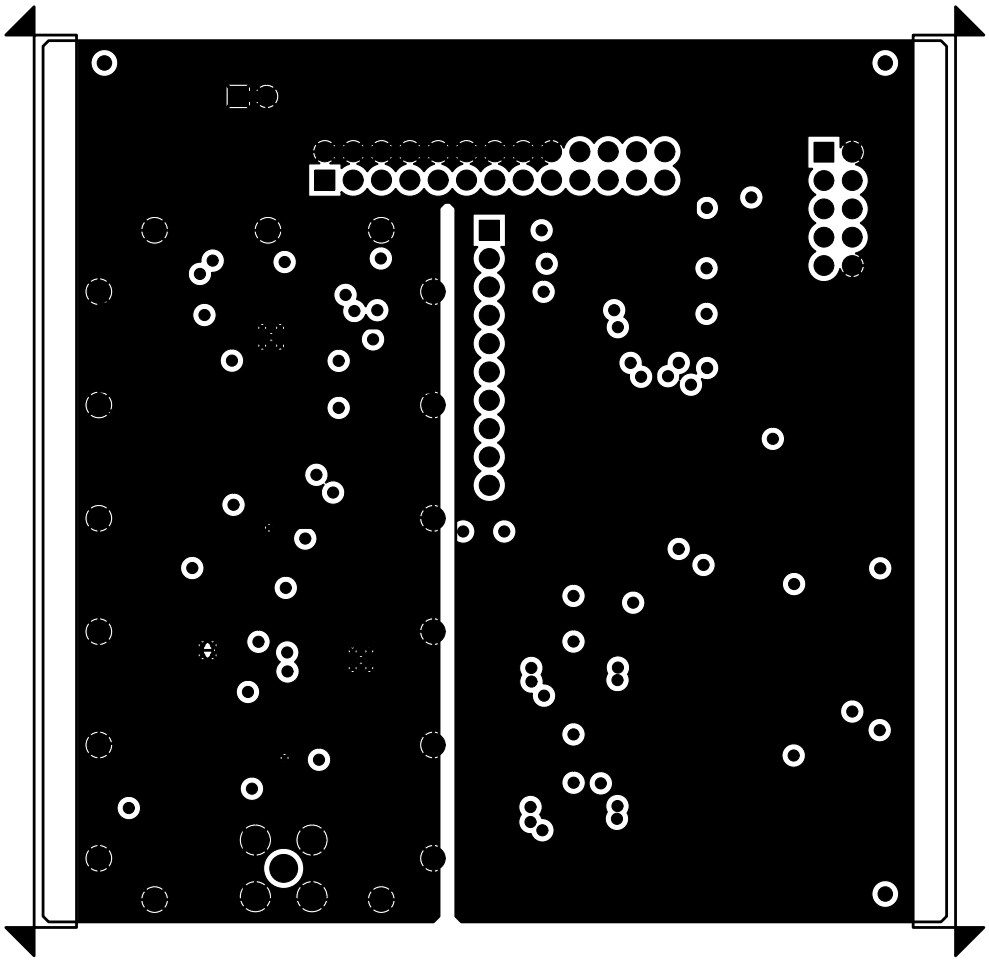

Figure F.9: PCB Ground Layer

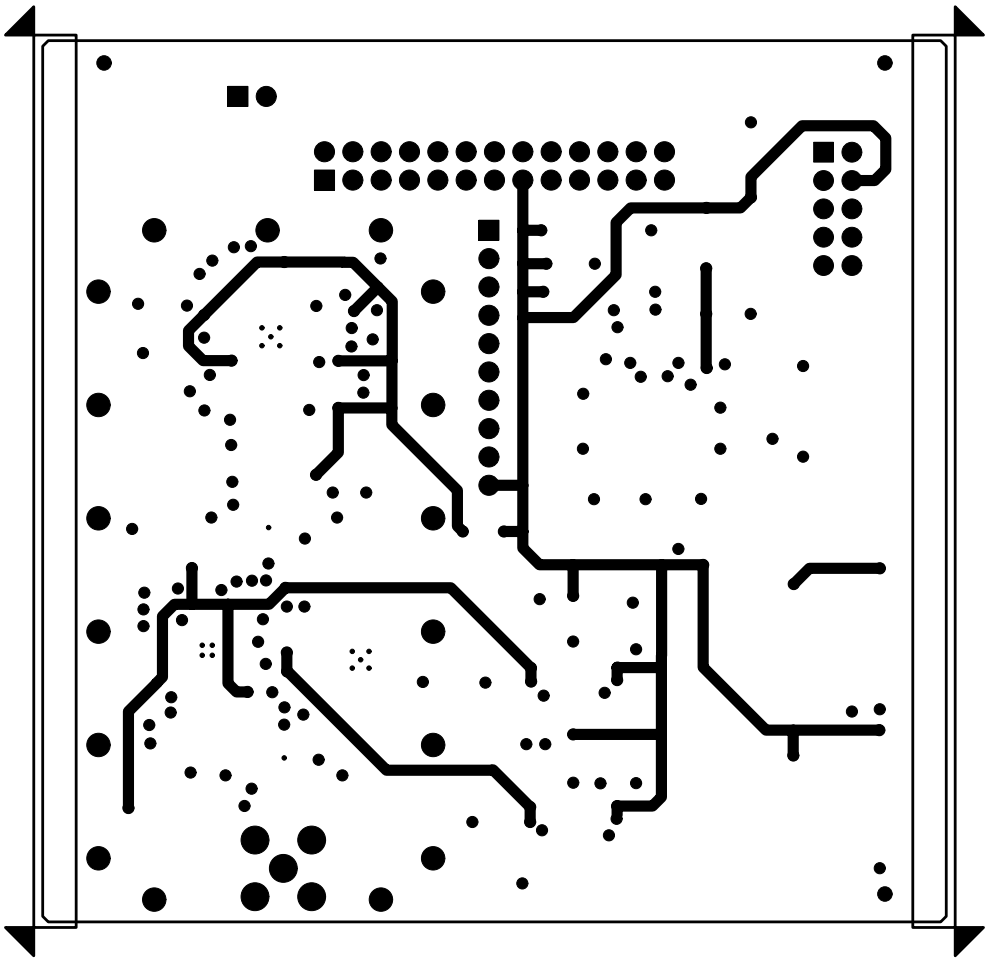

Figure F.10: PCB Power Layer

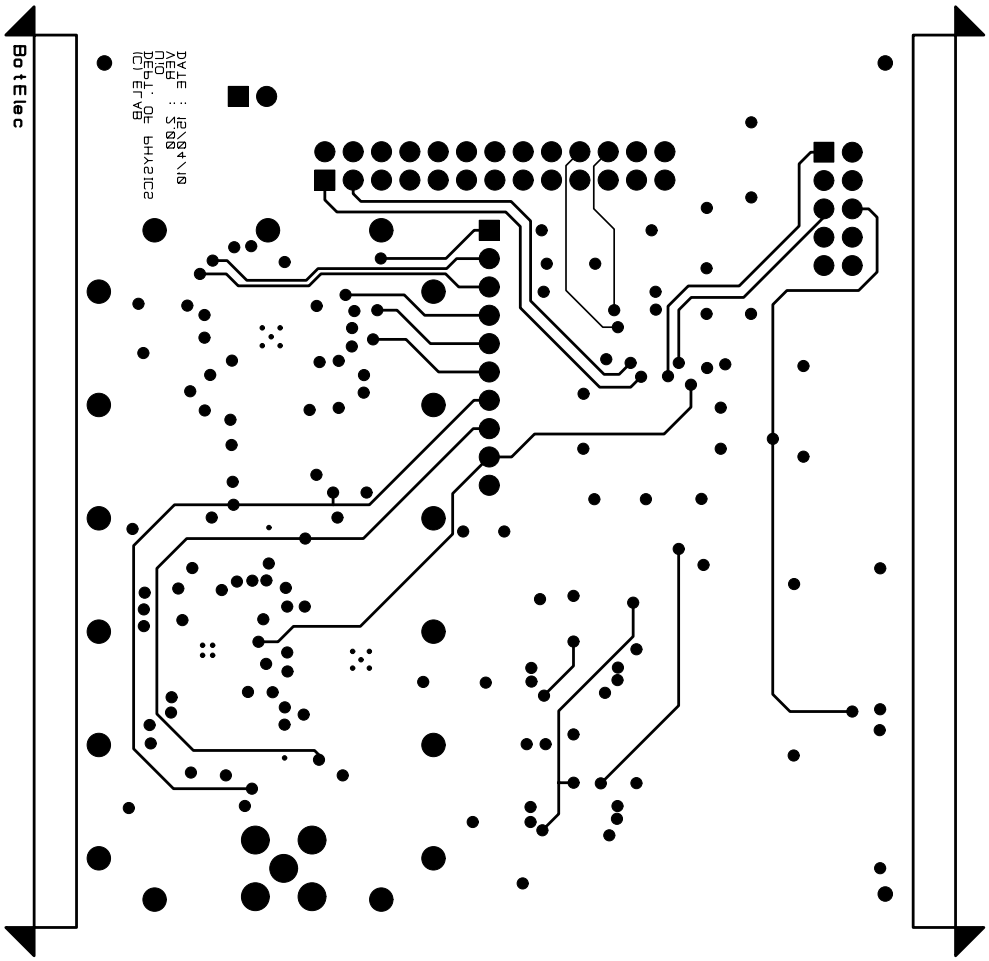

Figure F.11: PCB BotElec Layer

Appendix G

Source Code

This appendix contains the Hardware Abstraction Layer firmware packet HAL and the debug interface firmware packet.
The firmware has been written in C-code in AVR Studio 4 and compiled by GNU compiler. The code has been documented using the Doxygen software standard.

G.1 Hardware Abstraction Layer

./firmware/hal.h

```
4
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************************
 *
 * \brief
9 *      Hardware Abstraction Layer (HAL) driver module
 *
 *      This file contains the firmware drivers to operate the CubeSTAR
 *    communication system.
 *    The systems default mode is in receive mode, the variable
14 *    uint8_t new_data : true / false
 *    must be polled periodically to check for new available data. If true
 *    a new command has been receieved and can be retrieved from
 *    uint8_t *command_buffer.
 *
19 *    The drivers is developed to be able to transmitt a telemetry signal
 *    or beacon signal using:
 *    int8_t send_packet(uint8_t type_of_data, uint8_t buffer, bytes)
 *    uint8_t type_of_data : TELEMETRY/BEACON
 *    uint8_t *buffer :  < 64 bytes (TELEMETRY) / < 256 bytes (BEACON)
24 *    uint8_t bytes : number of bytes to transmit
 *
 *
 *
 *
29 * \par
 *
 *
 *
 * \author
34 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:      \n
39 *
```

```
 *
 ******************************************************************************/


#ifndef _HAL_
#define _HAL_

#define F_CPU 16000000UL

#include "avr\io.h"
#include "global_def.h"
#include "cc1101_library.h"
#include "cc1101.h"
#include "mcu.h"
#include "spi.h"
#include "comm.h"
#include "dac.h"
#include "usart.h"
#include "debug.h"
#include "rtc.h"
#include "beacon.h"


#define hal_init()  (mcu_init(), spi_init(), usart_init(), rtc_init(),
     system_default_mode())



#endif /* _HAL_ */
```

./firmware/comm.c

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file ********************************************************
 *
 * \brief
 * Communication driver source file.
 *
 * This file contains the communication function for the transceiver.
 *
 * \par
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:          \n
 ******************************************************************/


/*******************  INCLUDES  *****************/
#include "global_def.h"
#include "cc1101_library.h"
#include "cc1101.h"
#include "comm.h"
#include "usart.h"
#include "mcu.h"
#include "beacon.h"

/*******************   DEFINITIONS   *****************/
#define RX    1
#define TX    2
#define IDLE  3

#define TRX_IRQ_vector  PORTE_INT0_vect

#define TX_BUFFER_SIZE  63

/*********************  MACROS  *********************/


/*******************   FUNCTIONS   *******************/
int8_t send_telemetry(uint8_t *txBuffer, uint8_t bytes);
```

```
/******************  VARIABLES ******************/
volatile uint8_t rx_buffer_lenght;
volatile uint8_t trx_state = RX;



/*! \brief Calls the telemetry or beacon driver
int8_t send_packet(uint8_t type_of_data, uint8_t buffer, bytes)
*
*
* \param uint8_t type_of_data
* \param uint8_t buffer
* \param uint8_t bytes
*
*   \return Status code
*   \retval (uint8_t)OK
*   \retval (uint8_t)FAILED
*/
int8_t send_packet(uint8_t type_of_data, uint8_t *buffer, uint8_t bytes)
{

 /* atomic operation */
 cli();

 int8_t status = OK;

 switch(type_of_data)
 {
  case(BEACON):
   trx_state = TX;
   send_beacon(buffer, bytes);
  break;

  case(TELEMETRY):
   trx_state = TX;
   send_telemetry(buffer, bytes);
  break;

  default:
   status = FAILED;
  break;

 }

 /* reset system to default mode */
 system_default_mode();

 return OK;
}

/*! \brief Place communication system in RX mode
 *
 *
 *  \param
 *
 *  \return Status code
 *  \retval (uint8_t)OK
 *  \retval (uint8_t)FAILED
 */
uint8_t system_default_mode(void)
{
 /* atomic operation */
 cli();

  /* place radio front-end in RX mode */
 RX_MODE();
 _delay_ms(200);

 new_command = false;
 trx_state = RX;

  /* flush rx FIFO */
 cc1101_flush_rx_fifo();

 /* reset and load configuration into transceiver */
 cc1101_config();

 /* activate trx interrupt */
 activate_trx_irq();

 /* place transceiver in rx mode */
 cc1101_rx_mode();
```

```
 /* end atomic operation */
 clr_int0_flag();
 sei();

 return OK;
}


/*! \brief Transmitts a packet
int8_t send_telemetry(uint8_t *txBuffer, uint8_t bytes)
*
*
*   This function can be used to transmit a packet with packet
* length up to 63 bytes.
*   To use this function, GD00 must be configured to be asserted
* when sync word is sent and de-asserted at the end of the packet
* => halSpiWriteReg(CCxxx0_IOCFG0, 0x06);
*   The function implements polling of GDO0. First it waits for GD00
* to be set and then it waits for it to be cleared.
*
* \param uint8_t *txBuffer
* \param uint8_t bytes
*
*   \return Status code
*   \retval (uint8_t)OK
*   \retval (uint8_t)FAILED
*/
int8_t send_telemetry(uint8_t *txBuffer, uint8_t bytes)
{
 trx_state = TX;

 /* place radio front-end in TX mode */
 TX_MODE();
 _delay_ms(200);

 /* load configuration into transceiver */
 cc1101_config();

 /* flush tx FIFO */
 cc1101_flush_tx_fifo();

 /* upload data to transceiver */
 cc1101_burst_write_reg(CC1101_TXFIFO, txBuffer, TX_BUFFER_SIZE);


 /* transmitt data */
 cc1101 tx mode();



 return OK;

}


/*! \brief Interrupt service routine for transceiver
 *
 *  When initialized this ISR is called on every rising edge
 *  on GD0.
 */
ISR(PORTE_INT0_vect)
{



 switch(trx_state)
 {
  case(TX):
   /* if transmitting finished, do nothing */
   if(!Is_GDO0_PIN_high())
   {}

  /* if transmitting started, do nothing */
   if(Is GDO0 PIN high())
   {}

  break;

  case(RX):

   /*  FIFO filled, download data */
```

```
    if(!Is_GDO0_PIN_high())
    {

      rx_buffer_lenght = cc1101_read_reg(CC1101_RXFIFO);
219   cc1101_burst_read_reg(CC1101_RXFIFO, command_buffer, rx_buffer_lenght);
    new_command = true;


    }
224
    /* receiving data, do nothing */
    else
    {

229   }

    break;

    /* system recovery */
234 default:
     system_default_mode();
    break;
   }

239 }
```

./firmware/comm.h

```
  /* This file has been prepared for Doxygen automatic documentation generation.*/
  /*! \file *********************************************************
4  *
   * \brief
   *       COMM driver header file.
   *
   *       This file contains the function declarations and
9  *     definitions for the COMM driver.
   *
   * \par
   *
   * \author
14 *       Johan L. Tresvig \n
   *       email: j.l.tresvig@fys.uio.no
   *
   * $Revision:
   * $Date:      \n
19 *
   *********************************************************************/


  /*  DEFINITIONS */
24 #ifndef _TRANSMISSION_DRIVER_
  #define _TRANSMISSION_DRIVER_


  #define BEACON     1
29 #define TELEMETRY 2

  volatile uint8_t *command_buffer;
  volatile uint8_t new_command;

34 uint8_t system_default_mode(void);
  int8_t send_packet(uint8_t type_of_data, uint8_t *buffer, uint8_t bytes);




39

  #endif /* _TRANSMISSION_DRIVER_ */
```

./firmware/beacon.c

```
  /* This file has been prepared for Doxygen automatic documentation generation.*/
  /*! \file *********************************************************************
4  *
   * \brief
   *       BeaconI driver source file.
   *
   *       This file contains the function implementations  of the beacon driver.
```

```
 *
 *
 * \par
 *
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:          \n
 *
 *
 ******************************************************************************/


/******************  INCLUDES  *****************/
#include "global_def.h"
#include "beacon.h"
#include "cc1101.h"



/******************  DEFINITIONS *****************/

#define DOT           (1200/WPM)
#define DASH          (3*DOT)
#define INTER_SIG_DELAY   (1*DOT)
#define INTER_LETTER_DELAY  (3*DOT)
#define INTER_WORD_DELAY  (7*DOT)



/*********************  MACROS  *********************/



/******************   FUNCTIONS   *******************/
void send_dot(void);
void send_dash(void);
int8_t send_letter(uint8_t letter);



/******************  VARIABLES *****************/




/*! \brief Writes an ASCII string to a beacon signal,
 *
 *  This function writes an ASCIÏ string to .
 *
 *  \param uint8_t *beacon_string,
 *  \param uint8_t bytes
 *
 *  \return Status code
 *  \retval (uint8_t)OK / (uint8_t)FAILED
 */
int8_t send_beacon(uint8_t *beacon_string, uint8_t bytes)
{
 uint8_t i=0;
 uint8_t status = OK;


 while(i<bytes)
 {
  /* if, space between word */
  if(beacon_string[i]==' ') { _delay_ms(INTER_WORD_DELAY); }

  /* else, send letter */
  else
  {
   send_letter(beacon_string[i]);
```

```
94      /* signal end of letter */
        _delay_ms(INTER_LETTER_DELAY);
      }

      /* increase buffer cnt */
99      i++;
    }

    /* signal end of beacon */
    _delay_ms(INTER_WORD_DELAY);
104

    return status;
}

109

/*! \brief Transmitts a morse code "dot"
 *
 *  This function transmitts a "dot"
114 *
 *  \param uint8_t none
 *
 *  \return none
 */
119 void send_dot(void)
{
  cc1101_tx_mode();
  _delay_ms(DOT);
  cc1101_idle_mode();
124
}


/*! \brief Transmitts a morse code "dash"
129 *
 *  This function transmitts a "dash"
 *
 *  \param uint8_t none
 *
134 *  \return none
 */
void send_dash(void)
{
  cc1101_tx_mode();
139  _delay_ms(DASH);
  cc1101_idle_mode();

}

144


/*! \brief Writes an ASCII character to a beacon signal,
 *
 *  This function writes an ASCIÏ character to a morse code
149 *  signal.
 *
 *  \param uint8_t uint8_t letter
 *
 *  \return Status code
154 *  \retval (uint8_t)OK / (uint8_t)FAILED
 */
int8_t send_letter(uint8_t letter)
{
 int8_t status = OK;
159
 switch(letter)
 {
  case('A'):
   send_dot();
164   _delay_ms(INTER_SIG_DELAY);
   send_dash();
  break;

  case('B'):
169   send_dash();
    delay_ms(INTER_SIG_DELAY);
   send_dot();
    delay_ms(INTER_SIG_DELAY);
   send_dot();
174   _delay_ms(INTER_SIG_DELAY);
   send_dot();
  break;
```

```
    case('C'):
179   send_dash();
      _delay_ms(INTER_SIG_DELAY);
      send_dot();
      _delay_ms(INTER_SIG_DELAY);
      send_dash();
184     delay_ms(INTER_SIG_DELAY);
      send_dot();
    break;

    case('D'):
189   send_dash();
      _delay_ms(INTER_SIG_DELAY);
      send_dot();
      _delay_ms(INTER_SIG_DELAY);
      send_dot();
194 break;

    case('E'):
      send_dot();
    break;
199
    case('F'):
      send_dot();
      _delay_ms(INTER_SIG_DELAY);
      send_dot();
204   _delay_ms(INTER_SIG_DELAY);
      send_dash();
      _delay_ms(INTER_SIG_DELAY);
      send_dot();
    break;
209
    case('G'):
      send_dash();
      _delay_ms(INTER_SIG_DELAY);
      send_dash();
214   _delay_ms(INTER_SIG_DELAY);
      send_dot();
    break;

    case('H'):
219   send_dot();
      _delay_ms(INTER_SIG_DELAY);
      send_dot();
      _delay_ms(INTER_SIG_DELAY);
      send_dot();
224   _delay_ms(INTER_SIG_DELAY);
      send_dot();
    break;

    case('I'):
229   send_dot();
      _delay_ms(INTER_SIG_DELAY);
      send_dot();
    break;

234

    case('J'):
      send_dot();
      _delay_ms(INTER_SIG_DELAY);
      send_dash();
239   _delay_ms(INTER_SIG_DELAY);
      send_dash();
      _delay_ms(INTER_SIG_DELAY);
      send_dash();
    break;
244
    case('K'):
      send_dash();
      _delay_ms(INTER_SIG_DELAY);
      send_dot();
249   _delay_ms(INTER_SIG_DELAY);
      send_dash();
    break;


254 case('L'):
      send_dot();
      _delay_ms(INTER_SIG_DELAY);
      send_dash();
      _delay_ms(INTER_SIG_DELAY);
259   send_dot();
      _delay_ms(INTER_SIG_DELAY);
```

```
    send_dot();
    _delay_ms(INTER_LETTER_DELAY);
  break;

  case('M'):
   send_dash();
   _delay_ms(INTER_SIG_DELAY);
   send_dash();
  break;

  case('N'):
   send_dash();
   _delay_ms(INTER_SIG_DELAY);
   send_dot();
  break;

  case('O'):
   send_dash();
   _delay_ms(INTER_SIG_DELAY);
   send_dash();
   _delay_ms(INTER_SIG_DELAY);
   send_dash();
   _delay_ms(INTER_LETTER_DELAY);
  break;

  case('P'):
   send_dot();
   _delay_ms(INTER_SIG_DELAY);
   send_dash();
   _delay_ms(INTER_SIG_DELAY);
   send_dash();
   _delay_ms(INTER_SIG_DELAY);
   send_dot();
   _delay_ms(INTER_LETTER_DELAY);
  break;

  case('Q'):
   send_dash();
   _delay_ms(INTER_SIG_DELAY);
   send_dash();
   _delay_ms(INTER_SIG_DELAY);
   send_dot();
   _delay_ms(INTER_SIG_DELAY);
   send_dash();
  break;

  case('R'):
   send_dot();
   _delay_ms(INTER_SIG_DELAY);
   send_dash();
   _delay_ms(INTER_SIG_DELAY);
   send_dot();

  break;


  case('S'):
   send_dot();
   _delay_ms(INTER_SIG_DELAY);
   send_dot();
   _delay_ms(INTER_SIG_DELAY);
   send_dot();
  break;

  case('T'):
   send_dash();
  break;

  case('U'):
   send_dot();
   _delay_ms(INTER_SIG_DELAY);
   send_dot();
   _delay_ms(INTER_SIG_DELAY);
   send_dash();
   _delay_ms(INTER_LETTER_DELAY);
  break;

  case('V'):
   send_dot();
   _delay_ms(INTER_SIG_DELAY);
   send_dot();
   _delay_ms(INTER_SIG_DELAY);
   send_dot();
```

```
    _delay_ms(INTER_SIG_DELAY);
    send_dash();
   break;

   case('W'):
    send_dot();
     delay_ms(INTER_SIG_DELAY);
    send_dash();
    _delay_ms(INTER_SIG_DELAY);
    send_dash();
     _delay_ms(INTER_LETTER_DELAY);
   break;

   case('X'):
    send_dash();
    _delay_ms(INTER_SIG_DELAY);
    send_dot();
    _delay_ms(INTER_SIG_DELAY);
    send_dot();
    _delay_ms(INTER_SIG_DELAY);
    send_dash();
   break;

   case('Y'):
    send_dash();
     delay_ms(INTER_SIG_DELAY);
    send_dot();
    _delay_ms(INTER_SIG_DELAY);
    send_dash();
    _delay_ms(INTER_SIG_DELAY);
    send_dash();
   break;

   case('Z'):
    send_dash();
    _delay_ms(INTER_SIG_DELAY);
    send_dash();
    _delay_ms(INTER_SIG_DELAY);
    send_dot();
    _delay_ms(INTER_SIG_DELAY);
    send_dot();
   break;


   default:
    status = FAILED;
   break;

  }

  return status;
}
```

./firmware/beacon.h

```

/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************************
 *
 * \brief
 *       Beacon driver header file.
 *
 *       This file contains the function declarations and definitions for the
 *     beacon driver.
 *
 *
 * \par
 *
 *
 *
 * \author
 *       Johan L. Tresvig \n
 *       email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:       \n
 *
 *
 *****************************************************************************/
```

```
27
#ifndef _BEACON_DRIVER_
#define _BEACON_DRIVER_

32 /*  DEFINITIONS */

/* define Words per Minute */
#define WPM   6 /*!< \brief Define Words per Minute. */

37 #ifndef WPM
/* prevent compiler error by supplying a default */
# warning "Words per Minute not defined for beacon signal, WPM = 6"
# define WPM  6
#endif
42



int8_t send_beacon(uint8_t *beacon_string, uint8_t bytes);
47




52
#endif  /* _BEACON_DRIVER_ */
```

./firmware/cc1101.c

```
2 /* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file ********************************************************
 *
 * \brief
 * Chipcon CC1101 driver source file.
7 * This file contains the function implementations the CC1101 driver.
 *
 * \par
 *
 * \author
12 *       Johan L. Tresvig \n
 *       email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:          \n
17 *
 ********************************************************************************/



22 /*******************  INCLUDES  *****************/
#include "global def.h"
#include "mcu.h"
#include "cc1101_library.h"
#include "cc1101.h"
27 #include "spi.h"

/*******************  DEFINITIONS *******************/
#define FSK    0
#define GFSK   1
32 #define OOK    3
#define MODULATION  FSK

#define _1200_BPS 5
#define _9600_BPS 8
37 #define DATA_RATE _9600_BPS

#define PACKET_SIZE 61


42 S_RF_SETTINGS_T TRX_PACKET_MODE_2FSK_9600BPS = {
(0x06<<FREQ_IF_bp), // FSCTRL1   Frequency synthesizer control.
   (0x00<<FREQOFF_bp), // FSCTRL0   Frequency synthesizer control.
   (0x10<<FREQ_HIGH_bp), // FREQ2     Frequency control word, high byte.
   (0xBB<<FREQ_MIDDLE_bp), // FREQ1     Frequency control word, middle byte.
47   (0x13<<FREQ_LOW_bp),// FREQ0     Frequency control word, low byte.
   (3<<CHANBW_E_bp)|(3<<CHANBW_M_bp)|(DATA_RATE<<DRATE_E_bp),// MDMCFG4   Modem
        configuration.
   (0x83<<DRATE_M_bp), // MDMCFG3   Modem configuration.
     (1<<DEM_DCFILT_OFF_bp)|(MODULATION<<MOD_FORMAT_bp)|(0<<MANCHESTER_EN_bp)|(0<<
          SYNCH_MODE_bp), // MDMCFG2   Modem configuration.
```

```
    (0<<FEC_EN_bp)|(0<<NUM_PREAMBLE_bp)|(1<<CHANSPC_E_bp),// MDMCFG1    Modem
        configuration.
52  (0x46<<CHANSPC_M_bp),// MDMCFG0    Modem configuration.
    (0x00<<CHAN_bp),// CHANNR     Channel number.
    (1<<DEVIATION_E_bp)|(5<<DEVIATION_M_bp),// DEVIATN    Modem deviation setting
        (when FSK modulation is enabled).
    (3<<LNA_CURRENT_bp)|(0<<LNA2MIX_CURRENT_bp)|(1<<LODIV_BUF_CURRENT_RX_bp)|(2<<
        MIX_CURRENT_bp),    // FREND1     Front end RX configuration.
    (1<<LODIV_BUF_CURRENT_TX_bp)|(0<<PA_POWER_bp),// FREND0     Front end TX
        configuration.
57  (1<<FS_AUTOCAL_bp)|(2<<PO_TIMEOUT_bp)|(0<<PIN_CTRL_EN_bp)|(0<<
        XOSC_FORCE_ON_bp),// MCSM0      Main Radio Control State Machine
        configuration.
    (0<<FOC_BS_CS_GATE_bp)|(3<<FOC_PRE_K_bp)|(FOC_POST_K_bp)|(1<<FOC_LIMIT_bp),
        // FOCCFG     Frequency Offset Compensation Configuration.
    (0<<BS_PRE_KI_bp)|(1<<BS_PRE_KP_bp)|(1<<BS_POST_KI_bp)|(1<<BS_POST_KP_bp)
        |(0<<BS_LIMIT_bp),    // BSCFG      Bit synchronization Configuration.
    (3<<MAX_DVGA_GAIN_bp)|(0<<MAX_LNA_GAIN_bp)|(7<<MAGN_TARGET_bp),// AGCCTRL2
        AGC control.
    (0<<AGC_LNA_PRIORITY_bp)|(0<<CARRIER_SENSE_REL_THR_bp)|(0<<
        CARRIER_SENSE_ABS_THR_bp),// AGCCTRL1  AGC control.
62  (2<<HYST_LEVEL_bp)|(3<<WAIT_TIME_bp)|(0<<AGC_FREEZE_bp)|(2<<FILTER_LENGHT_bp)
        ,// AGCCTRL0  AGC control.
    (0xCA<<FSCAL3_bp)|(2<<CHP_CURR_CAL_EN_bp),// FSCAL3     Frequency synthesizer
        calibration.
    (1<<VCO_CORE_H_EN_bp)|(0x0A<<FSCAL2_bp),  // FSCAL2     Frequency synthesizer
        calibration.
    (0<<FSCAL1_bp),// FSCAL1     Frequency synthesizer calibration.
    (0x1F<<FSCAL0_bp),// FSCAL0     Frequency synthesizer calibration.
67  0x59,// FSTEST     Frequency synthesizer calibration.
    0x81,// TEST2      Various test settings.
    0x35,// TEST1      Various test settings.
    0x09,// TEST0      Various test settings.
    (1<<ADC_RETENTION_bp)|(0<<CLOSE_IN_RX_bp)|(0x0F<<FIFO_THR_bp),// FIFOTHR
        RXFIFO and TXFIFO thresholds.
72  (0<<GDO2_INV_bp)|(0x29<<GDO2_CFG_bp),// IOCFG2     GDO2 output pin
        configuration.
    (0<<TEMP_SENSOR_ENABLE_bp)|(0<<GDO0_INV_bp)|(0x06<<GDO0_CFG_bp),// IOCFG0
        GDO0 output pin configuration.
    (0<<ADR_CHK_bp),// PKTCTRL1  Packet automation control.
    (0<<WHITE_DATA_bp)|(0<<PKT_FORMAT_bp)|(0<<CRC_EN_bp)|(0<<LENGHT_CONFIG_bp),//
        PKTCTRL0  Packet automation control.
    (0x00<<DEVICE_ADDR_bp),// ADDR       Device address.
77  (PACKET_SIZE<<PACKET_LENGHT_bp)// PKTLEN     Packet length.
};

/********************  MACROS  ************************/
#define Is_cc1101_ready(x)  ((x < CHIP_RDYn) ? true:false)
82


/*******************   FUNCTIONS   *********************/
uint8_t cc1101_get_status(void);
void cc1101_flush_rx_fifo(void);
87 void cc1101_flush_tx_fifo(void);


/*******************  VARIABLES ********************/
uint8_t *data_buffer;
92




97


/*  STATUS READ     */
uint8_t cc1101_get_status(void){return cc1101_strobe_func(CC1101_SNOP);}

102 /*  COMMAND STROBES */
void cc1101_tx_mode(void){cc1101_strobe_func(CC1101_STX);}
void cc1101_rx_mode(void){cc1101_strobe_func(CC1101_SRX);}
void cc1101_idle_mode(void){cc1101_strobe_func(CC1101_SIDLE);}
void cc1101_flush_rx_fifo(void){cc1101_strobe_func(CC1101_SFRX);}
107 void cc1101_flush_tx_fifo(void){cc1101_strobe_func(CC1101_SFTX);}

/* Function need to be atomic,
*  the reset will cause a XOSC on the GDO0 pin which will
*  trigger the irq vector
112 */
void cc1101_reset_transceiver(void){
                    cc1101_strobe_func(CC1101_SRES);
                      cc1101_write_reg(CC1101_IOCFG0, 0x06);
                     }
117
```

```
/*! \brief Function transmit a strobe command to CC1101
 *
 *  This function transmitts a strobe command to CC1101
 *  on the SPI interface and returns the status byte from CC1101
 *
 *  \param uint8_t reg_addr
 *
 *  \return Status byte from CC1101
 *  \retval uint8_t
 */
uint8_t cc1101_strobe_func(uint8_t reg_addr)
{
 /* compile buffer */
 data_buffer[0] = CC1101_READ|CC1101_STROBE|reg_addr;

 /* transmitt header byte, return status value in data buffer[0] */
 spi_transceive_packet(data_buffer, 1);

 /* return status value of CC1101 */
 return data_buffer[0];
}


/*! \brief Function reads a register value from CC1101
 *
 *
 *  \param uint8_t reg_addr
 *
 *  \return Register value from CC1101
 *  \retval uint8_t
 */
uint8_t cc1101_read_reg(uint8_t reg_addr)
{
 /* compile buffer */
 uint8_t cmd_type = 0x00;

 /* if addresse < 0x30, register access */
 if(reg_addr < 0x30) {cmd_type = CC1101_RW_REG;}

 /* else if adresse is equal or larger than 0x30, status access */
 else if (reg_addr >= 0x30) {cmd_type = CC1101_STATUS;}

 data_buffer[0] = CC1101_READ|cmd_type|reg_addr;
 data_buffer[1] = DUMMY_BYTE;

 /* transmitt header byte and a dummy byte, returns register value in data buffer
     [2] */
 spi_transceive_packet(data_buffer, 2);

 /* return register value */
 return data_buffer[1];
}


/*! \brief Function writes to a register in CC1101
 *
 *
 *  \param uint8_t reg_addr
 *  \param uint8_t reg_value
 *
 *  \return dummy value
 *  \retval int8_t
 */
int8_t cc1101_write_reg(uint8_t reg_addr, uint8_t reg_value)
{
 /* compile buffer */
 data_buffer[0] = CC1101_WRITE|reg_addr;
 data_buffer[1] = reg_value;

 /* transmitt header byte, return register value in data buffer[1] */
 spi_transceive_packet(data_buffer, 2);

 /* return dummy value */
 return OK;
}


/*! \brief Function burst writes to a register in CC1101
 *
 *
 *  \param uint8_t reg_addr
```

```
 *   \param uint8_t reg_value
 *
 *   \return dummy value
 *   \retval int8_t
 */
int8_t cc1101_burst_write_reg(uint8_t reg_addr, uint8_t *data, uint8_t bytes)
{
 /* compile buffer */
 data_buffer = data;

 /* transmitt header byte and write data */
  spi_burst_transceive_packet(reg_addr, data_buffer, bytes, CC1101_WRITE);

 /* return dummy value */
 return OK;
}


/*! \brief Function burst reads from a register in CC1101
 *
 *
 *   \param uint8_t reg_addr
 *   \param uint8_t reg_value
 *
 *   \return dummy value
 *   \retval int8_t
 */
int8_t cc1101_burst_read_reg(uint8_t reg_addr, uint8_t *data, uint8_t bytes)
{

 /* transmitt header byte and read data */
  spi_burst_transceive_packet(reg_addr, data, bytes, CC1101_READ);

 /* return dummy value */
 return OK;

}


/*! \brief Function configures the CC1101
 *
 *  Function writes to the configuration register in CC110.
 *  The settings determines the setup of the transceiver
 *
 *   \param uint8_t struct S_RF_SETTINGS *cc1101_settings
 *
 *   \return dummy value
 *   \retval cc1101_state_t
 */
int8_t cc1101_init(S_RF_SETTINGS_T *cc1101_settings)
{

 /* Write register settings */
 cc1101_write_reg(CC1101_FSCTRL1,  cc1101_settings->FSCTRL1);
 cc1101_write_reg(CC1101_FSCTRL0,  cc1101_settings->FSCTRL0);
 cc1101_write_reg(CC1101_FREQ2,    cc1101_settings->FREQ2);
 cc1101_write_reg(CC1101_FREQ1,    cc1101_settings->FREQ1);
 cc1101_write_reg(CC1101_FREQ0,    cc1101_settings->FREQ0);
 cc1101_write_reg(CC1101_MDMCFG4,  cc1101_settings->MDMCFG4);
 cc1101_write_reg(CC1101_MDMCFG3,  cc1101_settings->MDMCFG3);
 cc1101_write_reg(CC1101_MDMCFG2,  cc1101_settings->MDMCFG2);
 cc1101_write_reg(CC1101_MDMCFG1,  cc1101_settings->MDMCFG1);
 cc1101_write_reg(CC1101_MDMCFG0,  cc1101_settings->MDMCFG0);
 cc1101_write_reg(CC1101_CHANNR,   cc1101_settings->CHANNR);
 cc1101_write_reg(CC1101_DEVIATN,  cc1101_settings->DEVIATN);
 cc1101_write_reg(CC1101_FREND1,   cc1101_settings->FREND1);
 cc1101_write_reg(CC1101_FREND0,   cc1101_settings->FREND0);
 cc1101_write_reg(CC1101_MCSM0 ,   cc1101_settings->MCSM0 );
 cc1101_write_reg(CC1101_FOCCFG,   cc1101_settings->FOCCFG);
 cc1101_write_reg(CC1101_BSCFG,    cc1101_settings->BSCFG);
 cc1101_write_reg(CC1101_AGCCTRL2, cc1101_settings->AGCCTRL2);
 cc1101_write_reg(CC1101_AGCCTRL1, cc1101_settings->AGCCTRL1);
 cc1101_write_reg(CC1101_AGCCTRL0, cc1101_settings->AGCCTRL0);
 cc1101_write_reg(CC1101_FSCAL3,   cc1101_settings->FSCAL3);
 cc1101_write_reg(CC1101_FSCAL2,   cc1101_settings->FSCAL2);
 cc1101_write_reg(CC1101_FSCAL1,   cc1101_settings->FSCAL1);
 cc1101_write_reg(CC1101_FSCAL0,   cc1101_settings->FSCAL0);
 cc1101_write_reg(CC1101_FSTEST,   cc1101_settings->FSTEST);
 cc1101_write_reg(CC1101_TEST2,    cc1101_settings->TEST2);
 cc1101_write_reg(CC1101_TEST1,    cc1101_settings->TEST1);
 cc1101_write_reg(CC1101_TEST0,    cc1101_settings->TEST0);
 cc1101_write_reg(CC1101_FIFOTHR,  cc1101_settings->FIFOTHR);
 cc1101_write_reg(CC1101_IOCFG2,   cc1101_settings->IOCFG2);
 cc1101_write_reg(CC1101_IOCFG0,   cc1101_settings->IOCFG0);
```

```
    cc1101_write_reg(CC1101_PKTCTRL1, cc1101_settings->PKTCTRL1);
    cc1101_write_reg(CC1101_PKTCTRL0, cc1101_settings->PKTCTRL0);
287 cc1101_write_reg(CC1101_ADDR,     cc1101_settings->ADDR);
    cc1101_write_reg(CC1101_PKTLEN,   cc1101_settings->PKTLEN);



292
    return OK;
}


297 /*! \brief Function writes a bit field to a register in CC1101
     *
     *  Function reads a register value from CC1101, clears the bit field
     *  and writes the new value to the reister in CC1101
     *
302  *  \param uint8_t reg_addr
     *  \param uint8_t bit_mask
     *  \param uint8_t bit_position
     *  \param uint8_t bit_value
     *
307  *  \return dummy value
     *  \retval int8_t
     */
int8_t cc1101_write_bitfield(uint8_t reg_addr, uint8_t bit_mask, uint8_t
    bit_position, uint8_t value)
{
312 /* get registyer value */
    uint8_t reg_value = cc1101_read_reg(reg_addr);

    /* clear bitfield and input new value */
    reg_value = (reg_value & ~bit_mask) | (value<<bit_position);
317
    /* write new value to register*/
    cc1101_write_reg(reg_addr, reg_value);

    return OK;
322 }


/*! \brief Function uploades a configuration to the transceiver
     *
327  *  Function reads a register value from CC1101, clears the bit field
     *  and writes the new value to the reister in CC1101
     *
     *  \param none
     *
332  *  \return none
     *  \retval
     */
void cc1101_config(void)
{
337 /* reset transceiver */
    cc1101_strobe_func(CC1101_SRES);

    /* upload configuration */
    cc1101_init(&TRX_PACKET_MODE_2FSK_9600BPS);
342 }
```

./firmware/cc1101.h

```
2 /* This file has been prepared for Doxygen automatic documentation generation.*/
  /*! \file ******************************************************
    *
    * \brief
    *       Transceiver driver header file.
7   *
    *       This file contains the function declarations and
    *    definitions for the CC1101 driver.
    *
    * \par
12  *
    * \author
    *       Johan L. Tresvig \n
    *       email: j.l.tresvig@fys.uio.no
    *
17  * $Revision:
    * $Date:     \n
    *
    ***************************************************************/
```

```
22
#ifndef _CC1101_DRIVER_
#define _CC1101_DRIVER_

27 /*  DEFINITIONS */
/* define trx port to use, default port is PORTE */
#define TRX_PORT  PORTE /*!< \brief Define data port. */

#define GDO2_PIN_bm   PIN2_bm
32 #define GDO0_PIN_bm   PIN3_bm
#define Is_GDO0_PIN_high()   ((TRX_PORT.IN & GDO0_PIN_bm) ? true:false)
#define Is_GDO2_PIN_high()   ((TRX_PORT.IN & GDO2_PIN_bm) ? true:false)

#define CMD_PACKET        0x01
37 #define DATA_PACKET       0x02
#define OTHER_TRANSMISSION  0
#define BURST_TRANSMISSION  0x40



42

//
    ------------------------------------------------------------------------------------------------------


/*! \brief RF_SETTINGS is a data structure which contains all relevant CC1101
     registers. */
47 typedef struct S_RF_SETTINGS{
    uint8_t FSCTRL1;   /*!< \brief Frequency synthesizer control. */
    uint8_t FSCTRL0;   /*!< \brief Frequency synthesizer control. */
    uint8_t FREQ2;     /*!< \brief Frequency control word, high byte. */
    uint8_t FREQ1;     /*!< \brief Frequency control word, middle byte. */
52  uint8_t FREQ0;     /*!< \brief Frequency control word, low byte. */
    uint8_t MDMCFG4;   /*!< \brief Modem configuration. */
    uint8_t MDMCFG3;   /*!< \brief Modem configuration. */
    uint8_t MDMCFG2;   /*!< \brief Modem configuration. */
    uint8_t MDMCFG1;   /*!< \brief Modem configuration. */
57  uint8_t MDMCFG0;   /*!< \brief Modem configuration. */
    uint8_t CHANNR;    /*!< \brief Channel number. */
    uint8_t DEVIATN;   /*!< \brief Modem deviation setting (when FSK modulation
         is enabled). */
    uint8_t FREND1;    /*!< \brief Front end RX configuration. */
    uint8_t FREND0;    /*!< \brief Front end RX configuration. */
62  uint8_t MCSM0;     /*!< \brief Main Radio Control State Machine configuration
         . */
    uint8_t FOCCFG;    /*!< \brief Frequency Offset Compensation Configuration.
         */
    uint8_t BSCFG;     /*!< \brief Bit synchronization Configuration. */
    uint8_t AGCCTRL2;  /*!< \brief AGC control. */
  uint8_t AGCCTRL1;  /*!< \brief AGC control. */
67  uint8_t AGCCTRL0;  /*!< \brief AGC control. */
    uint8_t FSCAL3;    /*!< \brief Frequency synthesizer calibration. */
    uint8_t FSCAL2;    /*!< \brief Frequency synthesizer calibration. */
  uint8_t FSCAL1;    /*!< \brief Frequency synthesizer calibration. */
    uint8_t FSCAL0;    /*!< \brief Frequency synthesizer calibration. */
72  uint8_t FSTEST;    /*!< \brief Frequency synthesizer calibration control */
    uint8_t TEST2;     /*!< \brief Various test settings. */
    uint8_t TEST1;     /*!< \brief Various test settings. */
    uint8_t TEST0;     /*!< \brief Various test settings. */
    uint8_t FIFOTHR;   /*!< \brief RXFIFO and TXFIFO thresholds. */
77  uint8_t IOCFG2;    /*!< \brief GDO2 output pin configuration. */
    uint8_t IOCFG0;    /*!< \brief GDO0 output pin configuration */
    uint8_t PKTCTRL1;  /*!< \brief Packet automation control. */
    uint8_t PKTCTRL0;  /*!< \brief Packet automation control. */
    uint8_t ADDR;      /*!< \brief Device address. */
82  uint8_t PKTLEN;    /*!< \brief Packet length. */
} S_RF_SETTINGS_T;



87



int8_t cc1101_init(S_RF_SETTINGS_T *cc1101_settings);
uint8_t cc1101_strobe_func(uint8_t reg_addr);
92 uint8_t cc1101_read_reg(uint8_t reg_addr);
int8_t cc1101_write_reg(uint8_t reg_addr, uint8_t reg_value);
int8_t cc1101_burst_write_reg(uint8_t reg_addr, uint8_t *data, uint8_t bytes);
int8_t cc1101_burst_read_reg(uint8_t reg_addr, uint8_t *data, uint8_t bytes);
int8_t cc1101_write_bitfield(uint8_t reg_addr, uint8_t bit_mask, uint8_t
     bit_position, uint8_t value);
97
```

```
void cc1101_config(void);
void cc1101_reset_transceiver(void);
void cc1101_tx_mode(void);
void cc1101_rx_mode(void);
void cc1101_idle_mode(void);

void cc1101_flush_rx_fifo(void);
void cc1101_flush_tx_fifo(void);

#endif /* _CC1101_DRIVER_ */
```

./firmware/cc1101_library.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file ********************************************************
 *
 * \brief
 * Register definition file for Texas instrument CC1101 transceiver.
 *
 * \par
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:      \n
 *
 *
 ********************************************************************/


#ifndef _CC110_REG_DEF_
#define _CC110_REG_DEF_

/*  DEFINITIONS */
#define CC101_BURST       0x40
#define CC1101_STROBE     0x00
#define CC1101_STATUS     0x40
#define CC1101_RW_REG     0x00

#define CC1101_READ       0x80
#define CC1101_WRITE      0x00

//  ------------------------------------------------------------------------------------------------

/*  Control registers */
#define CC1101_IOCFG2         0x00        /*!< \brief GDO2 output pin configuration. */
#define CC1101_IOCFG1         0x01        /*!< \brief GDO1 output pin configuration. */
#define CC1101_IOCFG0         0x02        /*!< \brief GDO0 output pin configuration. */
#define CC1101_FIFOTHR        0x03        /*!< \brief GDO2 output pin configuration. */
#define CC1101_SYNC1          0x04        /*!< \brief Sync word, high byte. */
#define CC1101_SYNC0          0x05        /*!< \brief Sync word, low byte. */
#define CC1101_PKTLEN         0x06        /*!< \brief Packet length. */
#define CC1101_PKTCTRL1       0x07        /*!< \brief Packet automation control. */
#define CC1101_PKTCTRL0       0x08        /*!< \brief Packet automation control. */
#define CC1101_ADDR           0x09        /*!< \brief Device address. */
#define CC1101_CHANNR         0x0A        /*!< \brief Channel number. */
#define CC1101_FSCTRL1        0x0B        /*!< \brief Frequency synthesizer control. */
#define CC1101_FSCTRL0        0x0C        /*!< \brief Frequency synthesizer control. */
#define CC1101_FREQ2          0x0D        /*!< \brief Frequency control word, high byte. */
#define CC1101_FREQ1          0x0E        /*!< \brief Frequency control word, middle byte. */
#define CC1101_FREQ0          0x0F        /*!< \brief Frequency control word, low byte. */
#define CC1101_MDMCFG4        0x10        /*!< \brief Modem configuration. */
#define CC1101_MDMCFG3        0x11        /*!< \brief Modem configuration. */
#define CC1101_MDMCFG2        0x12        /*!< \brief Modem configuration. */
#define CC1101_MDMCFG1        0x13        /*!< \brief Modem configuration. */
#define CC1101_MDMCFG0        0x14        /*!< \brief Modem configuration. */
#define CC1101_DEVIATN        0x15        /*!< \brief Modem deviation setting. */
```

```
#define CC1101_MCSM2          0x16          /*!< \brief Main Radio Control State
    Machine configuration. */
#define CC1101_MCSM1          0x17          /*!< \brief Main Radio Control State
    Machine configuration. */
#define CC1101_MCSM0          0x18          /*!< \brief Main Radio Control State
    Machine configuration. */
#define CC1101_FOCCFG         0x19          /*!< \brief Frequency Offset Compensation
     configuration. */
#define CC1101_BSCFG          0x1A          /*!< \brief Bit Synchronization
    configuration. */
#define CC1101_AGCCTRL2       0x1B          /*!< \brief AGC control. */
#define CC1101_AGCCTRL1       0x1C          /*!< \brief AGC control. */
#define CC1101_AGCCTRL0       0x1D          /*!< \brief AGC control. */
#define CC1101_WOREVT1        0x1E          /*!< \brief High byte Event 0 timeout. */
#define CC1101_WOREVT0        0x1F          /*!< \brief Low byte Event 0 timeout. */
#define CC1101_WORCTRL        0x20          /*!< \brief Wake On Radio control. */
#define CC1101_FREND1         0x21          /*!< \brief Front end RX configuration.
    */
#define CC1101_FREND0         0x22          /*!< \brief Front end TX configuration.
    */
#define CC1101_FSCAL3         0x23          /*!< \brief Frequency synthesizer
    calibration. */
#define CC1101_FSCAL2         0x24          /*!< \brief Frequency synthesizer
    calibration. */
#define CC1101_FSCAL1         0x25          /*!< \brief SFrequency synthesizer
    calibration. */
#define CC1101_FSCAL0         0x26          /*!< \brief Frequency synthesizer
    calibration. */
#define CC1101_RCCTRL1        0x27          /*!< \brief RC oscillator configuration.
    */
#define CC1101_RCCTRL0        0x28          /*!< \brief RC oscillator configuration.
    */
#define CC1101_FSTEST         0x29          /*!< \brief Frequency synthesizer
    calibration control. */
#define CC1101_PTEST          0x2A          /*!< \brief Production test. */
#define CC1101_AGCTEST        0x2B          /*!< \brief AGC test. */
#define CC1101_TEST2          0x2C          /*!< \brief Various test settings. */
#define CC1101_TEST1          0x2D          /*!< \brief Various test settings. */
#define CC1101_TEST0          0x2E          /*!< \brief Various test settings. */

/* Strobe commands  */
#define CC1101_SRES           0x30          /*!< \brief Reset chip. */
#define CC1101_SFSTXON        0x31          /*!< \brief Enable and calibrate
    frequency synthesizer (if MCSM0.FS_AUTOCAL=1). If in RX/TX: Go to a wait
    state where only the synthesizer is running (for quick RX / TX turnaround).
    */
#define CC1101_SXOFF          0x32          /*!< \brief Turn off crystal oscillator.
    */
#define CC1101_SCAL           0x33          /*!< \brief Calibrate frequency
    synthesizer and turn it off (enables quick start). */
#define CC1101_SRX            0x34          /*!< \brief Enable RX. Perform
    calibration first if coming from IDLE and MCSM0.FS AUTOCAL=1. */
#define CC1101_STX            0x35          /*!< \brief In IDLE state: Enable TX.
    Perform calibration first if MCSM0.FS_AUTOCAL=1. If in RX state and CCA is
    enabled: Only go to TX if channel is clear. */

#define CC1101_SIDLE          0x36          /*!< \brief Exit RX / TX, turn off
    frequency synthesizer and exit Wake-On-Radio mode if applicable. */
#define CC1101_SAFC           0x37          /*!< \brief Perform AFC adjustment of the
     frequency synthesizer */
#define CC1101_SWOR           0x38          /*!< \brief Start automatic RX polling
    sequence (Wake-on-Radio) */
#define CC1101_SPWD           0x39          /*!< \brief Enter power down mode when
    CSn goes high. */
#define CC1101_SFRX           0x3A          /*!< \brief Flush the RX FIFO buffer. */
#define CC1101_SFTX           0x3B          /*!< \brief Flush the TX FIFO buffer. */
#define CC1101_SWORRST        0x3C          /*!< \brief Reset real time clock. */
#define CC1101_SNOP           0x3D          /*!< \brief No operation. May be used to
    pad strobe commands to two bytes for simpler software. */


/*  Status registers  */
#define CC1101_PARTNUM          0x30  /*!< \brief Chip part number. */
#define CC1101_VERSION          0x31  /*!< \brief Chip version number. */
#define CC1101_FREQEST          0x32  /*!< \brief Frequency offset estimation of
    the carrier. */
#define CC1101_LQI              0x33  /*!< \brief Demodulator estimate for link
    quality. */
#define CC1101_RSSI             0x34  /*!< \brief Received signal sthrenght
    indicator. */
#define CC1101_MARCSTATE        0x35  /*!< \brief Main radio state machine state.
    */
#define CC1101_WORTIME1         0x36  /*!< \brief High byte of WOR Time. */
#define CC1101_WORTIME0         0x37  /*!< \brief Low byte of WOR Time. */
```

```
#define CC1101_PKTSTATUS          0x38  /*!< \brief Current GDOx status and Packet
    status. */
#define CC1101_VCO_VC_DAC         0x39  /*!< \brief Current stting from PLL
    Calibration module. */
#define CC1101_TXBYTES            0x3A  /*!< \brief Underflow and Number of bytes.
    */
#define CC1101_RXBYTES            0x3B  /*!< \brief Overflow and Number of bytes.
    */
#define CC1101_RCCTRL1_STATUS     0x3C  /*!< \brief Last RC Oscillator Calibration
    Result. */
#define CC1101_RCCTRL0_STATUS     0x3D  /*!< \brief Last RC Oscillator Calibration
    Result. */

#define CC1101_PATABLE            0x3E  /*!< \brief Various test settings. */
#define CC1101_TXFIFO             0x3F  /*!< \brief Various test settings. */
#define CC1101_RXFIFO             0x3F  /*!< \brief Various test settings. */


/*  STATUS BYTE  */
#define CHIP_RDYn 0x80


/*  IOCFG2 - GDO2 Output Pin Configuration  */
 #define GDO2_INV_bm  0x40
 #define GDO2_CFG_bm  0x1F

 #define GDO2_INV_bp  6
 #define GDO2_CFG_bp  0

/*  IOCFG1 - GDO1 Output Pin Configuration  */
 #define GDO_DS_bm    0x80
 #define GDO1_INV_bm  0x40
 #define GDO1_CFG_bm  0x1F

 #define GDO_DS_bp    7
 #define GDO1_INV_bp  6
 #define GDO1_CFG_bp  0

/*  IOCFG0 - GDO0 Output Pin Configuration  */
 #define TEMP_SENSOR_ENABLE_bm  0x80
 #define GDO0_INV_bm       0x40
 #define GDO0_CFG_bm       0x1F

 #define TEMP_SENSOR_ENABLE_bp  7
 #define GDO0_INV_bp       6
 #define GDO0_CFG_bp       0


/*  FIFOTHR - RX FIFO and TX FIFO Thresholds  */
 #define ADC_RETENTION_bm 0x40
 #define CLOSE_IN_RX_bm   0x30
 #define FIFO_THR_bm      0x0F

 #define ADC_RETENTION_bp 6
 #define CLOSE_IN_RX_bp   4
 #define FIFO_THR_bp      0


/*  PKTLEN - Packet Lenght  */
 #define PACKET_LENGHT_bm 0xFF
 #define PACKET_LENGHT_bp 0


/*  PKTCTRL1 - Packet Automation Control  */
 #define PQT_bm         0xE0
 #define CRC_AUTOFLUSH_bm 0x08
 #define APPEND_STATUS_bm 0x04
 #define ADR_CHK_bm       0x03

 #define PQT_bp         5
 #define CRC_AUTOFLUSH_bp 3
 #define APPEND_STATUS_bp 2
 #define ADR_CHK_bp       0


/*  PKTCTRL0 - Packet Automation Control  */
 #define WHITE_DATA_bm     0x40
 #define PKT_FORMAT_bm     0x30
 #define CRC_EN_bm         0x04
 #define LENGHT_CONFIG_bm 0x03

 #define WHITE_DATA_bp     6
```

```
 #define PKT_FORMAT_bp     4
 #define CRC_EN_bp         2
 #define LENGHT_CONFIG_bp  0

/*  ADDR - Device Address */
 #define DEVICE_ADDR_bm    0xFF
 #define DEVICE_ADDR_bp    0

/*  CHANNR - Channel Number */
 #define CHAN_bm      0xFF
 #define CHAN_bp      0

/*  FSCTRL1 - Frequency Synthesizer Control */
 #define FREQ_IF_bm       0x1F
 #define FREQ_IF_bp       0

/*  FSCTRL0 - Frequency Synthesizer Control */
 #define FREQOFF_bm       0xFF
 #define FREQOFF_bp       0

/*  FREQ2 - Frequency Control Word, High Byte */
 #define FREQ_HIGH_bm     0x3F
 #define FREQ_HIGH_bp     0

/*  FREQ2 - Frequency Control Word, High Byte */
 #define FREQ_MIDDLE_bm    0xFF
 #define FREQ_MIDDLE_bp    0

/*  FREQ2 - Frequency Control Word, High Byte */
 #define FREQ_LOW_bm      0xFF
 #define FREQ_LOW_bp      0

/*  MDMCFG4 - Modem Configuration */
 #define CHANBW_E_bm      0xC0
 #define CHANBW_M_bm      0x30
 #define DRATE_E_bm       0x0F

 #define CHANBW_E_bp      6
 #define CHANBW_M_bp      4
 #define DRATE_E_bp       0

/*  MDMCFG3 - Modem Configuration */
 #define DRATE_M_bm       0xFF
 #define DRATE_M_bp       0

/*  MDMCFG2 - Modem Configuration */
 #define DEM_DCFILT_OFF_bm  0x80
 #define MOD_FORMAT_bm      0x70
 #define MANCHESTER_EN_bm   0x08
 #define SYNCH_MODE_bm      0x07

 #define DEM_DCFILT_OFF_bp  7
 #define MOD_FORMAT_bp      4
 #define MANCHESTER_EN_bp   3
 #define SYNCH_MODE_bp      0

/*  MDMCFG1 - Modem Configuration */
 #define FEC_EN_bm         0x80
 #define NUM_PREAMBLE_bm   0x70
 #define CHANSPC_E_bm      0x03

 #define FEC_EN_bp         7
 #define NUM_PREAMBLE_bp   4
 #define CHANSPC_E_bp      0

/*  MDMCFG0 - Modem Configuration */
 #define CHANSPC_M_bm       0xFF
 #define CHANSPC_M_bp       0

/*  DEVIATN - Modem Deviation Setting */
 #define DEVIATION_E_bm     0x70
 #define DEVIATION_M_bm     0x07
```

```
#define DEVIATION_E_bp    4
#define DEVIATION_M_bp    0

/* MCSM2 - Main Radio Control State Machine Configuration */
#define RX_TIME_RSSI_bm   0x10
#define RX_TIME_QUAL_bm   0x80
#define RX_TIME_bm        0x07

#define RX_TIME_RSSI_bp   4
#define RX_TIME_QUAL_bp   3
#define RX_TIME_bp        0

/* MCSM1 - Main Radio Control State Machine Configuration */
#define CCA_MODE_bm       0x30
#define RXOFF_MODE_bm     0x0C
#define TXOFF_MODE_bm     0x03

#define CCA_MODE_bp       4
#define RXOFF_MODE_bp     2
#define TXOFF_MODE_bp     0

/* MCSM0 - Main Radio Control State Machine Configuration */
#define FS_AUTOCAL_bm     0x30
#define PO_TIMEOUT_bm     0x0C
#define PIN_CTRL_EN_bm    0x02
#define XOSC_FORCE_ON_bm  0x01

#define FS_AUTOCAL_bp     4
#define PO_TIMEOUT_bp     2
#define PIN_CTRL_EN_bp    1
#define XOSC_FORCE_ON_bp  0


/* FOCCFG - Frequency Offset Compensation Configuration */
#define FOC_BS_CS_GATE_bm  0x20
#define FOC_PRE_K_bm       0x18
#define FOC_POST_K_bm      0x04
#define FOC_LIMIT_bm       0x03

#define FOC_BS_CS_GATE_bp  5
#define FOC_PRE_K_bp       3
#define FOC_POST_K_bp      2
#define FOC_LIMIT_bp       0

/* BSCFG - Bit Synchronization Configuration */
#define BS_PRE_KI_bm       0xC0
#define BS_PRE_KP_bm       0x30
#define BS_POST_KI_bm      0x80
#define BS_POST_KP_bm      0x40
#define BS_LIMIT_bm        0x03

#define BS_PRE_KI_bp       6
#define BS_PRE_KP_bp       4
#define BS_POST_KI_bp      3
#define BS_POST_KP_bp      2
#define BS_LIMIT_bp        0

/* AGCCTRL2 - AGC Control */
#define MAX_DVGA_GAIN_bm   0xC0
#define MAX_LNA_GAIN_bm    0x38
#define MAGN_TARGET_bm     0x07

#define MAX_DVGA_GAIN_bp   6
#define MAX_LNA_GAIN_bp    3
#define MAGN_TARGET_bp     0

/* AGCCTRL1 - AGC Control */
#define AGC_LNA_PRIORITY_bm      0x40
#define CARRIER_SENSE_REL_THR_bm 0x30
#define CARRIER_SENSE_ABS_THR_bm 0x0F

#define AGC_LNA_PRIORITY_bp      6
#define CARRIER_SENSE_REL_THR_bp 4
#define CARRIER_SENSE_ABS_THR_bp 0
```

```
358 /*  AGCCTRL0 - AGC Control  */
    #define HYST_LEVEL_bm     0xC0
    #define WAIT_TIME_bm    0x30
    #define AGC_FREEZE_bm     0x0C
    #define FILTER_LENGHT_bm 0x03
363
    #define HYST_LEVEL_bp     6
    #define WAIT_TIME_bp    4
    #define AGC_FREEZE_bp     2
    #define FILTER_LENGHT_bp 0
368

    /*  WOREVT1 - High Byte Event0 Timeout  */
    #define EVENT0_HIGH_bm    0xFF
    #define EVENT0_HIGH_bp    0
373

    /*  WOREVT0 - Low Byte Event0 Timeout */
    #define EVENT0_LOW_bm     0xFF
    #define EVENT0_LOW_bp     0
378

    /*  WORCTRL - Wake On Radio Control */
    #define RC_PD_bm      0x80
    #define EVENT1_bm       0x60
383 #define RC_CAL_bm       0x08
    #define WOR_RES_bm      0x03

    #define RC_PD_bp      7
    #define EVENT1_bp       4
388 #define RC_CAL_bp       3
    #define WOR_RES_bp      0


    /*  FREND1 - Front End RX Configuration */
393 #define LNA_CURRENT_bm        0xC0
    #define LNA2MIX_CURRENT_bm      0x30
    #define LODIV_BUF_CURRENT_RX_bm  0xC0
    #define MIX_CURRENT_bm        0x03

398 #define LNA_CURRENT_bp        6
    #define LNA2MIX_CURRENT_bp      4
    #define LODIV_BUF_CURRENT_RX_bp  2
    #define MIX_CURRENT_bp        0

403
    /*  FREND0 - Front End TX Configuration */
    #define LODIV_BUF_CURRENT_TX_bm  0x30
    #define PA_POWER_bm          0x07

408 #define LODIV_BUF_CURRENT_TX_bp  4
    #define PA_POWER_bp          0


    /*  FSCAL3 - Frequency Synthesizer Calibration  */
413 #define FSCAL_bm      0xC0
    #define CHP_CURR_CAL_EN_bm 0x30
    #define FSCAL3_bm       0x0F

    #define FSCAL_bp      6
418 #define CHP_CURR_CAL_EN_bp 4
    #define FSCAL3_bp       0


    /*  FSCAL2 - Frequency Synthesizer Calibration  */
423 #define VCO_CORE_H_EN_bm 0x40
    #define FSCAL2_bm       0x1F

    #define VCO_CORE_H_EN_bp 5
    #define FSCAL2_bp       0
428

    /*  FSCAL1 - Frequency Synthesizer Calibration  */
    #define FSCAL1_bm       0x3F
    #define FSCAL1_bp       0
433

    /*  FSCAL1 - Frequency Synthesizer Calibration  */
    #define FSCAL0_bm       0x7F
    #define FSCAL0_bp       0
438

    /*  RCCTRL1 - RC Oscillator Configuration */
    #define RCCTRL1_bm      0x7F
```

```
 #define RCCTRL1_bp      0

/*  RCCTRL0 - RC Oscillator Configuration */
 #define RCCTRL0_bm      0x7F
 #define RCCTRL0_bp      0


/*  TEST0  Various Test Settings  */
 #define TEST0_bm      0xFD
 #define VCO_SEL_CAL_EN_bm  0x02


#endif /*  CC110 REG DEF  */
```

./firmware/dac.c

```


/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file ********************************************************************
 *
 * \brief
 *      DAC driver source file.
 *
 *      This file contains the DAC function for the XMEGA MCU
 *
 *
 * \par
 *
 *
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:          \n
 *
 *
 ********************************************************************************/

/******************  INCLUDES  *****************/
#include "global_def.h"
#include "dac.h"


/******************  DEFINITIONS *****************/

#define CH0EN           DAC_CH0EN_bm
#define CH1EN           DAC_CH1EN_bm

#define CH0             0
#define CH1             1

#define SINGLE_CHANNEL      DAC_CHSEL_SINGLE_gc
#define DUAL_CHANNEL        DAC_CHSEL_DUAL_gc

#define INTERNAL_VREF       DAC_REFSEL_INT1V_gc
#define AVCC_VREF           DAC_REFSEL_AVCC_gc


#define DAC_REFRESH_RATE        DAC_REFRESH_32CLK_gc
#define DAC_REFRESH_RATE_OFF    DAC_REFRESH_OFF_gc

#define DAC_CONV_INTERVAL    DAC_CONINTVAL_4CLK_gc



/*******************  MACROS  *******************/




/******************    FUNCTIONS    ******************/
```

```
int8_t dac_channel_write( volatile DAC_t * dac, uint16_t data, uint8_t channel );



/******************  VARIABLES ******************/



/*! \brief Initialize DAC module,
 *
 *  This function initializes the hardware on the MCU required to operate the
 *    transceiver system.
 *
 *  \param
 *
 *  \return Status code
 *  \retval (uint8_t)OK / (uint8_t)FAILED
 */
int8_t dac_init(void)
{

 /* load DAC CTRL register */
 DAC_MODULE.CTRLB = DUAL_CHANNEL;

 DAC_MODULE.CTRLC = AVCC_VREF;

 DAC_MODULE.TIMCTRL = DAC_REFRESH_RATE|DAC_CONV_INTERVAL;



 /* Enable DAC */
 DAC_MODULE.CTRLA = CH1EN|CH0EN;
 dac_enable();


 return OK;

}



/*! \brief  Write data to selected channel.
 *
 *  This function writes data to the selected channel data register. The program
 *  should wait for the data register to be ready if necessary. This ensures
 *  that no intermediate values are lost with very high update rates.
 *
 *  \note  The data must be in the format currently configured for the DAC,
 *         right or left adjusted.
 *
 *  \param  dac      Pointer to DAC module register section.
 *  \param  data     Data to be converted.
 *  \param  channel Selected channel in the DAC module, either CH0 or CH1.
 *
 *  \return Status code
 *  \retval (uint8_t)OK / (uint8_t)FAILED
 */
int8_t dac_channel_write( volatile DAC_t * dac, uint16_t data, uint8_t channel )
{
  if ( channel == CH0 ) {
    dac->CH0DATA = data;
  }
  else{
    dac->CH1DATA = data;
  }


 return OK;
}



/*! \brief  Sets analog value onto TX_GAIN_SIGNAL.
 *
 *  This function writes an analog value in the range
 *  from 0 to 3V onto the TX_GAIN_SIGNAL to the HPA.
 *
```

```
 *  \note
 *
 *  \param  dac      Pointer to DAC module register section.
 *  \param  data     Data to be converted.
 *  \param  channel Selected channel in the DAC module, either CH0 or CH1.
 *
 *  \return Status code
 *  \retval (uint8_t)OK / (uint8_t)FAILED
 */
int8_t set_dac_output(uint16_t voltage)
{return dac_channel_write(&DAC_MODULE, voltage, CH1);}
```

./firmware/dac.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************************
 *
 * \brief
 *      XMEGA DAC driver header file.
 *
 *      This file contains the function declarations and definitions for the
 *    XMEGA DAC driver.
 *
 *
 * \par
 *
 *
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:      \n
 *
 *
 *****************************************************************************/


#ifndef _DAC_DRIVER_
#define _DAC_DRIVER_


/*  DEFINITIONS */

/* define DAC module to use, default module is DACB */
#define DAC_MODULE  DACB  /*!< \brief Define DAC port. */

#ifndef DAC_MODULE
/* prevent compiler error by supplying a default */
# warning "DAC_MODULE not defined for DAC interface, DACB set as default"
# define DAC_MODULE DACB
#endif


/* values referenced to AVCC = 3.00V, value = ( Vout * 0xFFF) / AVCC */
#define DAC_OFFSET  200          /* offset value, most be adjusted */
#define _2V6     3549-(DAC_OFFSET)
#define _2V7     3685-(DAC_OFFSET)
#define _2V8     3822-(DAC_OFFSET)
#define _2V9     3958-(DAC_OFFSET)
#define _3V0     4095-(DAC_OFFSET)

#define ENABLE       DAC_ENABLE_bm
#define dac_enable()   DAC_MODULE.CTRLA |= ENABLE;
#define dac_disable()    DAC_MODULE.CTRLA &= ~ENABLE;

int8_t dac_init(void);


int8_t set_dac_output(uint16_t voltage);




#endif  /* _DAC_DRIVER_ */
```

./firmware/mcu.c

```
3
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file ********************************************************************
 *
 * \brief
8 *      MCU driver source file.
 *
 *      This file contains the IO and signal functions for the MCU.
 *
 *
13 * \par
 *
 *
 *
 * \author
18 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:          \n
23 *
 *
 ******************************************************************************/


28 /******************  INCLUDES  *****************/
#include "global_def.h"
#include "spi.h"
#include "mcu.h"
#include "cc1101.h"
33
#include "dac.h"
//#include "clksys_driver.h"



38
/******************  DEFINITIONS *****************/



43


/*********************  MACROS  *********************/

48



/******************   FUNCTIONS   ******************/
53 int8_t trx_irq_init(void);
int8_t sys_clk_init(void);
int8_t io_init(void);


58

/******************  VARIABLES *****************/



63



/*! \brief Initialize MCU,
68 *
 *  This function initializes the hardware on the MCU required to operate the
 *    transceiver system.
 *
 *  \param
 *
73 *  \return Status code
 *  \retval (uint8_t)OK / (uint8_t)FAILED
 */
int8_t mcu_init(void)
{
78 trx_irq_init();
 sys_clk_init();
 io_init();
```

```
83
    /* Configure RX interrupt from transceiver */
    // ConfigureInterrupt0(CLK_PIN, PORT_INT0LVL_HI_gc);

    return OK;
88
}


/*! \brief Configures IO for user interface.
93  *
    *  This function configures IO for TT&C user interface
    *
    *  \param none
    *
98  *  \return Status code
    *  \retval (uint8_t)OK / (uint8_t)FAILED
    */
int8_t io_init(void)
{
103
    /* LEDs */
    LED_PORT.DIRSET |= LED1_bm|LED2_bm;
    LED1_OFF();
    LED2_OFF();
108
    /* POWER*/
    POWER_PORT.DIRSET |= LNA_PWR_bm|HPA_PWR_bm;
    HPA_PWR_OFF();
    LNA_PWR_OFF();
113


    /* TX/RX MODE */
    TX_RX_CTRL_PORT.DIRSET |= RX_TX_MODE_1_bm|RX_TX_MODE_2_bm;
118 RX_MODE();


    SW_PORT.DIRSET  |= SWITCH1_bm|SWITCH2_bm;

123


    return OK;
}

128




/*! \brief Configures interrupt 0 for PORTE.
133 *
    *  This function configures interrupt 0 to be associated with a set of pins and
    *  sets the desired interrupt level.
    *
    *  \param none
138 *
    *  \return Status code
    *  \retval (uint8_t)OK / (uint8_t)FAILED
    */
int8_t trx_irq_init(void)
143 {
    /* trigger on both edges */
    TRX_PORT.PIN0CTRL = PORT_ISC_BOTHEDGES_gc;

    /* medium level pin irq */
148


    /* GD0 as trigger */
    TRX_PORT.INT0MASK = GDO0_PIN_bm;

153     /* enable medium irq */
    enable_med_level_irq();


    return OK;
158 }



/*! \brief Configures system clock for XMEGA.
163 *
    *  This function configures the system clock for the device
    *
```

```
 *   \param none
 *
 *   \return Status code
 *   \retval (uint8_t)OK / (uint8_t)FAILED
 */
int8_t sys_clk_init(void)
{

    /* set osc. range from 12 to 16MHZ, osc. source: XTAL */
        OSC.XOSCCTRL = OSC_FRQRANGE_12TO16_gc | OSC_XOSCSEL_XTAL_16KCLK_gc;

    /* enable ext. oscillator */
    OSC.CTRL |= OSC_XOSCEN_bm;

    /* wait for ext oscillator to become stable */
        while ( (OSC.STATUS & OSC_XOSCRDY_bm) == 0);

    /* open protected IO area */
    CCP=0xD8;

    /* select the ext. clock source as system clock */
        CLK.CTRL = CLK_SCLKSEL_XOSC_gc;

    /* switch off internal clock source */
        OSC.CTRL &= ~OSC_RC2MEN_bm;

    return OK;
}
```

./firmware/mcu.h

```

/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************************
 *
 * \brief
 *      XMEGA IO / IRQ header file.
 *
 *      This file contains the function declarations and definitions for the
 *    XMEGA IO and interrupt driver.
 *
 *
 * \par
 *
 *
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:      \n
 *
 *
 *****************************************************************************/


#ifndef _MCU_DRIVER_
#define _MCU_DRIVER_



/* define switch port to use, default port is PORTK */
#define SW_PORT PORTK /*!< \brief Define switch port. */

#define SWITCH1_bm  PIN1_bm
#define SWITCH2_bm  PIN2_bm

#define Is_SW1_pressed()  ((SW_PORT.IN & SWITCH1_bm) ? false:true)
#define Is_SW2_pressed()  ((SW_PORT.IN & SWITCH2_bm) ? false:true)

/* define led port to use, default port is PORTK */
#define LED_PORT  PORTK /*!< \brief Define led port. */

#define LED1_bm   PIN4_bm
#define LED2_bm   PIN3_bm

#define LED1_ON()    (LED_PORT.OUTCLR = LED1_bm)
#define LED1_OFF()     (LED_PORT.OUTSET = LED1_bm)
```

```
#define LED2_ON()    (LED_PORT.OUTCLR = LED2_bm)
#define LED2_OFF()     (LED_PORT.OUTSET = LED2_bm)
#define strobe_LED1() (LED_PORT.OUT ^= LED1_bm)
#define strobe_LED2() (LED_PORT.OUT ^= LED2_bm)



/* define POWER port to use, default port is PORTJ */
#define POWER_PORT  PORTJ /*!< \brief Define power port. */

#define HPA_PWR_bm  PIN0_bm
#define LNA_PWR_bm  PIN2_bm

#define LNA_PWR_ON()   (POWER_PORT.OUTCLR = LNA_PWR_bm)
#define LNA_PWR_OFF()  (POWER_PORT.OUTSET = LNA_PWR_bm)
#define HPA_PWR_ON()   (POWER_PORT.OUTCLR = HPA_PWR_bm)
#define HPA_PWR_OFF()  (POWER_PORT.OUTSET = HPA_PWR_bm)


/* define TX_RX_CTRL port to use, default port is PORTF */
#define TX_RX_CTRL_PORT PORTF /*!< \brief Define tx/rx ctrl port. */

#define RX_TX_MODE_1_bm PIN1_bm
#define RX_TX_MODE_2_bm PIN0_bm

#define RX_MODE()  (LNA_PWR_OFF(), HPA_PWR_OFF(), TX_RX_CTRL_PORT.OUTSET =
     RX_TX_MODE_1_bm, TX_RX_CTRL_PORT.OUTCLR = RX_TX_MODE_2_bm, LNA_PWR_ON())
#define TX_MODE()  (LNA_PWR_OFF(), HPA_PWR_OFF(), TX_RX_CTRL_PORT.OUTCLR =
     RX_TX_MODE_1_bm, TX_RX_CTRL_PORT.OUTSET = RX_TX_MODE_2_bm), HPA_PWR_ON()


/* test functions */

#define clr_int0_flag() (TRX_PORT.INTFLAGS |= PORT_INT0IF_bm)

#define enable_low_level_irq() (PMIC.CTRL |= PMIC_LOLVLEN_bm)
#define enable_med_level_irq() (PMIC.CTRL |= PMIC_MEDLVLEN_bm)

#define activate_trx_irq()     (clr_int0_flag(),TRX_PORT.INTCTRL |=
     PORT_INT0LVL_MED_gc)
#define deactivate_trx_irq()   (TRX_PORT.INTCTRL &= ~0x03, clr_int0_flag())

int8_t mcu_init(void);




#endif  /*  MCU_DRIVER  */
```

./firmware/spi.c

```

/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************************
 *
 * \brief
 *      XMEGA SPI driver source file.
 *
 *      This file contains the function implementations the XMEGA SPI driver.
 *
 *
 * \par
 *
 *
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:          \n
 *
 *
 *****************************************************************************/


/*******************  INCLUDES  *****************/
#include "global_def.h"
```

```
#include "spi.h"
#include "cc1101_library.h"
#include "cc1101.h"
31



/******************  DEFINITIONS *****************/
36
/*  IO pins */
#define SPI_CS_PIN     PIN4_bm    /*!< \brief Define IO pin for CS signal. */
#define SPI_MOSI_PIN   PIN5_bm    /*!< \brief Define IO pin for MOSI signal. */
#define SPI_SCLK_PIN   PIN7_bm    /*!< \brief Define IO pin for SCLK signal. */
41 /*  MISO predefined, ref. table 20-1, p.230 XMEGA datasheet */

/* SPIx.CTRL */

/*  SPIx.STATUS */
46 #define SPI_FLAG  SPI_IF_bm    /*!< \brief Define SPI trx-flag. */



/********************  MACROS  ********************/
51
/*  SPIx.CTRL */
#define spi_x2_SCLK_on()  (SPI_MODULE.CTRL |= SPI_CLK2X_bm)
#define spi_x2_SCLK_off()  (SPI_MODULE.CTRL &= ~SPI_CLK2X_bm)
#define spi_enable()     (SPI_MODULE.CTRL |= SPI_ENABLE_bm)
56 #define spi_disable()    (SPI_MODULE.CTRL &= ~SPI_ENABLE_bm)
#define spi_data_order(x) ((SPI_MODULE.CTRL &= ~SPI_DORD_bm), (SPI_MODULE.CTRL |=
      x))
#define spi_set_state(x)     ((SPI_MODULE.CTRL &= ~SPI_MASTER_bm), (SPI_MODULE.
    CTRL |= x))
#define spi_set_slave()
#define spi_set_mode(x)    ((SPI_MODULE.CTRL &= ~SPI_MODE_gm), (SPI_MODULE.CTRL |=
      x))
61 #define spi_SCLK_prescaler(x) ((SPI_MODULE.CTRL &= ~SPI_PRESCALER_gm), (
    SPI_MODULE.CTRL |= x))


/* SPIx.STATUS */

66 /*! \brief Check if new data is available.
 *
 *  \param
 *
 *  \return True if data available, false if not.
71  */
#define Is_new_byte_trx()      ((SPI_MODULE.STATUS &  0x80) ? true:false)


/*   SPIx IO   */
76
/*! \brief Place CS pin low.
 *
 *  \param
 *
81  *  \return
 */
#define spi_CS_pin_low()     (SPI_PORT.OUT &= ~0x10)

/*! \brief Place CS pin low.
86  *
 *  \param
 *
 *  \return
 */
91 #define spi_CS_pin_high()    (SPI_PORT.OUT |= 0x10)



/******************   FUNCTIONS   ******************/
96 uint8_t spi_transceive_byte(uint8_t data);
int8_t spi_initialize(uint8_t interface_state, uint8_t data_order, uint8_t
    data_mode, uint8_t clk_prescaler, uint8_t irq);



101 /******************  VARIABLES *****************/



106 int8_t spi_init(void)
```

```
{
 spi_initialize(SPI_MASTER, MSB, SPI_MODE_0, SPI_PRESCALER_DIV64, SPI_INTLVL0);

 return OK;
}

/*! \brief Initialize SPI module, SPIC/SPID/SPIE/SPIF defined in spi_driver.h
 *
 *  This function initializes a SPI module as master. The CTRL and INTCTRL
 *  registers for the SPI module is set according to the inputs to the function.
 *  In addition, data direction for the MOSI, CS and SCK pins is set to output.
 *
 *  \param uint8_t interface_state    SPI interface master or slave   (MASTER)
 *  \param uint8_t data_order     MSB or LSB shifted out first  (MSB)
 *  \param uint8_t data_mode      determine SCLK level and phase  (MODE 0)
 *  \param uint8_t clk_prescaler    SPI clock, division of Peripheral clock (
     SPI_PRESCALER_DIV2,4,8,16,32,64,128)
 *  \param uint8_t irq          SPI interrupt (SPI_INTLVL0-3, 0 = disabled)
 *
 *  \return Status code
 *  \retval (uint8_t)OK
 *  \retval (uint8_t)FAILED
 */
int8_t spi_initialize(uint8_t interface_state, uint8_t data_order, uint8_t
    data_mode, uint8_t clk_prescaler, uint8_t irq)
{

 if( interface_state == SPI_SLAVE ) {return FAILED;}

 /* Set CS, MOSI and SCLK as output, ref. XMEGA manual p.230 */
 SPI_PORT.DIRSET |= SPI_MOSI_PIN;
 SPI_PORT.DIRSET |= SPI_SCLK_PIN;
 SPI_PORT.DIRSET |= SPI_CS_PIN;
 spi_CS_pin_high();      /* place CS high  */


 /* load SPIx.CTRL register */
 SPI_MODULE.CTRL = interface_state|data_order|data_mode|clk_prescaler;

 /* load SPIx.INTCTRL */
 SPI_MODULE.INTCTRL |= irq;

 spi_enable();


 return OK;

}



/*! \brief Tranceives a packet on the SPI interface
 *
 *  This function transmitts a number of elements given by
 *  uint8_t bytes from spi_trx_buffer[] on the SPI bus
 *  The data received from slave is placed in spi_trx_buffer[]
 *
 *  \param uint8_t *spi_trx_buffer
 *  \param uint8_t bytes
 *
 *  \return Status code
 *  \retval (uint8_t)OK
 *  \retval (uint8_t)FAILED
 */
int8_t spi_transceive_packet(uint8_t *spi_trx_buffer, uint8_t bytes)
{

 uint8_t i = 0;

 spi_CS_pin_low();  /* Place CS low */
 _delay_us(750);  /* se datasheet for transceiver CC1101 */


 do
 {
   /*  Send byte */
   spi_trx_buffer[i] = spi_transceive_byte(spi_trx_buffer[i]);

   i++; /* Increment counter  */
```

```
 } while ( (i < bytes) );  //&& (!(spi_trx_buffer[0] & CHIP_RDYn))
 /* while more bytes to transceive && /CHIP_RDYn signal in statusbyte from slave
      is low*/

 spi_CS_pin_high(); /* Place CS pin high  */

 if((i == bytes)){return OK;} /* If all bytes transceived, return OK  */

 else{return FAILED;}      /* else, return FAILED  */

}



/*! \brief Tranceives a packet on the SPI interface
 *
 *  This function transmitts a number of elements given by
 *  uint8_t bytes from spi_trx_buffer[] on the SPI bus
 *  The data received from slave is placed in spi_trx_buffer[]
 *
 *  \param uint8_t *spi_trx_buffer
 *  \param uint8_t bytes
 *
 *  \return Status code
 *  \retval (uint8_t)OK
 *  \retval (uint8_t)FAILED
 */
int8_t spi_burst_transceive_packet(uint8_t reg_addr, uint8_t *spi_trx_buffer,
    uint8_t data_bytes, uint8_t RW)
{

 uint8_t i = 0;

 spi_CS_pin_low();  /* Place CS low */
 _delay_us(750);  /* se datasheet for transceiver CC1101 */

 spi_transceive_byte(reg_addr|RW|BURST_TRANSMISSION);


 do
 {
   /*  Send byte */
   spi_trx_buffer[i] = spi_transceive_byte(spi_trx_buffer[i]);

   i++; /* Increment counter  */

 } while ( (i < data_bytes) );  //&& (!(spi_trx_buffer[0] & CHIP_RDYn))
 /* while more bytes to transceive && /CHIP_RDYn signal in statusbyte from slave
      is low*/

 spi_CS_pin_high(); /* Place CS pin high  */

 if((i == data_bytes)){return OK;}  /* If all bytes transceived, return OK  */

 else{return FAILED;}      /* else, return FAILED  */

}


/*! \brief Tranceives a byte on the SPI interface
 *
 *  This function transmitts a byte on the SPI bus
 *  and returns the byte received from the slave
 *
 *  \param uint8_t data
 *
 *  \return SPIx.DATA
 *  \retval (uint8_t)
 */
uint8_t spi_transceive_byte(uint8_t data)
{
 /* Send new byte */
 SPI_MODULE.DATA = data;

 /* Wait for byte to be shifted out/in  */
 while( !Is_new_byte_trx() ) {}

 /* Return new byte */
 return SPI_MODULE.DATA;
}
```

./firmware/spi.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************************
 *
 * \brief
 *      XMEGA SPI driver header file.
 *
 *      This file contains the function declarations and definitions for the
 *    XMEGA SPI driver.
 *
 *
 * \par
 *
 *
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:      \n
 *
 *
 *****************************************************************************/



#ifndef _SPI_DRIVER_
#define _SPI_DRIVER_


/*  DEFINITIONS */

/* define SPI module to use, default module is SPIE */
#define SPI_MODULE  SPIE  /*!< \brief define SPI module. */
#define SPI_PORT  PORTE /*!< \brief Define SPI port. */



#define LSB             SPI_DORD_bm     /*!< \brief Bit mask for SPIx.CTRL. */
#define MSB             0x00        /*!< \brief Bit mask for SPIx.CTRL. */

#define SPI_MASTER          SPI_MASTER_bm   /*!< \brief Bit mask for SPIx.CTRL. */
#define SPI_SLAVE       0x00        /*!< \brief Bit mask for SPIx.CTRL. */

#define SPI_MODE_0          SPI_MODE_0_gc   /*!< \brief Bit mask for SPIx.CTRL. */
#define SPI_MODE_1          SPI_MODE_1_gc   /*!< \brief Bit mask for SPIx.CTRL. */
#define SPI_MODE_2          SPI_MODE_2_gc   /*!< \brief Bit mask for SPIx.CTRL. */
#define SPI_MODE_3          SPI_MODE_3_gc   /*!< \brief Bit mask for SPIx.CTRL. */

#define SPI_PRESCALER_DIV2    0x80          /*!< \brief Bit mask for SPIx.CTRL. */
#define SPI_PRESCALER_DIV4    0x00          /*!< \brief Bit mask for SPIx.CTRL. */
#define SPI_PRESCALER_DIV8    0x81          /*!< \brief Bit mask for SPIx.CTRL. */
#define SPI_PRESCALER_DIV16   0x01          /*!< \brief Bit mask for SPIx.CTRL. */
#define SPI_PRESCALER_DIV32   0x82          /*!< \brief Bit mask for SPIx.CTRL. */
#define SPI_PRESCALER_DIV64   0x02          /*!< \brief Bit mask for SPIx.CTRL. */
#define SPI_PRESCALER_DIV128  0x03          /*!< \brief Bit mask for SPIx.CTRL. */

#define SPI_INTLVL0         0x00        /*!< \brief Bit mask for SPIx.INTCTRL. */
#define SPI_INTLVL1         0x01        /*!< \brief Bit mask for SPIx.INTCTRL. */
#define SPI_INTLVL2         0x02        /*!< \brief Bit mask for SPIx.INTCTRL. */
#define SPI_INTLVL3         0x03        /*!< \brief Bit mask for SPIx.INTCTRL. */



int8_t spi_init(void);

int8_t spi_transceive_packet(uint8_t *spi_trx_buffer, uint8_t bytes);

int8_t spi_burst_transceive_packet(uint8_t reg_addr, uint8_t *spi_trx_buffer,
    uint8_t data_bytes, uint8_t RW);


#endif /* _SPI_DRIVER_ */
```

./firmware/rtc.h

```
4
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************************
 *
 * \brief
 *      XMEGA Real time counter driver header file.
 *
 *      This file contains the function declarations and definitions for the
 *    XMEGA real time counter driver.
14 *
 *
 * \par
 *
 *
19 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
24 * $Revision:
 * $Date:      \n
 *
 *
 *********************************************************************************/
29


#ifndef _REAL_TIME_COUNTER_DRIVER_
#define _REAL_TIME_COUNTER_DRIVER_
34

/*  DEFINITIONS */


39 #define rtc_enable()        (CLK.RTCCTRL |= CLK_RTCEN_bm)
#define rtc_disable()       (CLK.RTCCTRL &= ~CLK_RTCEN_bm)
#define rtc_prescale_div1024()      (RTC.CTRL |= RTC_PRESCALER_DIV1024_gc)
#define rtc_prescale_div1()     (RTC.CTRL |= RTC_PRESCALER_DIV1_gc)
#define rtc_comp_irq_en()     (RTC.INTCTRL |= RTC_COMPINTLVL_LO_gc)
44 #define rtc_set_comp_value(x) (RTC.COMP = x)

#define rtc_clr_cnt()         (RTC.CNTL = 0x00, RTC.CNTH = 0x00)


49 #define rtc_start()         (rtc_clr_cnt(),rtc_enable())
#define rtc_stop()          (rtc_disable(),rtc_clr_cnt())
#define rtc_init()          (rtc_prescale_div1(),rtc_clr_cnt())


54 #define Is_rtc_5sec()       ((RTC.CNT > 5120) ? true:false)
#define Is_rtc_10sec()        ((RTC.CNT > 10240) ? true:false)

#endif  /* _REAL_TIME_COUNTER_DRIVER_ */
```

./firmware/global_def.h

```
3 /* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************************
 *
 * \brief
 *      Global definitions header file.
8 *
 *      This file contains the generic global definitions for the HAL firmware
 *    package.
 *
 *
13 * \par
 *
 *
 *
 * \author
18 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:      \n
```

```
23  *
    *
    ******************************************************************************/


28
    #ifndef _GLOBAL_DEF_
    #define _GLOBAL_DEF_

    #include "avr\io.h"
33  #include "avr\interrupt.h"
    #include "util\delay.h"


38  #define system_reset()      (CCP = CCP_IOREG_gc, RST.CTRL = RST_SWRST_bm)

    #define true  1
    #define false 0

43
    #define FAILED     -1
    #define UNKNOWN    0
    #define OK         1

48
    #define DUMMY_BYTE  0x55    /*!< \brief Define a dummy byte. */



53

    #endif /* _GLOBAL_DEF_ */
```

G.2 Debug Interface

./firmware/debug.c

```
5   /* This file has been prepared for Doxygen automatic documentation generation.*/
    /*! \file ********************************************************************
    *
    * \brief
    *       Debug driver source file.
10  *
    *       This file contains the debug function for the XMEGA MCU
    *
    *
    * \par
15  *
    *
    *
    * \author
    *       Johan L. Tresvig \n
20  *       email: j.l.tresvig@fys.uio.no
    *
    * $Revision:
    * $Date:          \n
    *
25  *
    ******************************************************************************/


    /*******************  INCLUDES  *****************/
30  #include "hal.h"


    /*******************  DEFINITIONS *****************/
35


    /*********************  MACROS  *******************/

40
```

```
/*******************    FUNCTIONS    *******************/


45



/*******************  VARIABLES ******************/
uint8_t *buffer = "0123456789 0123456789 0123456789 0123456789 0123456789
     012345678";
50
uint8_t TEST72;
uint8_t TEST77;

uint8_t reset = false;
55




60
/*! \brief  Debug interface control function.
 *
 *  The function is called whenever a '#' is received on the UART interface.
 *  The function decodes the command call cmd(xx) and executes command
65 *
 *
 *  \param  none
 *
 *  \return Status code
70 *  \retval (uint8_t)OK / (uint8_t)FAILED
 */
int8_t command_call(void)
{
 //cli();
75 int8_t status = OK;
 uint8_t i=0;
 uint8_t cmd = 0;
 uint8_t cmd_tab[] = "cmd";
 uint8_t rx_buffer[6] = {};
80 uint8_t *tx_buffer = "invalid cmd                          ";

 while((i<5) && (status == OK))
 {
  /* wait for UART data to be sent from user */
85  while(!Is_USART_data_received()){}

  /* receive byte */
  rx_buffer[i] = usart_read_byte();

90   /* echo byte */
  usart_write_byte(rx_buffer[i]);

  /* check 3 first byte = 'c', 'm', 'd' */
  if(i<3)
95  {
   if(rx_buffer[i] != cmd_tab[i])
   {
    status = FAILED;
   }
100 }


  /* wait */
  //_delay_ms(200);
105 i++;
 }


 /* get spesific command */
110 cmd |= (10*(rx_buffer[3]-0x30)) + (rx_buffer[4]-0x30);

 if(status == OK)
 {
  switch(cmd)
115  {

/* USER INTERFACE */
   case(5):
    LED1_ON();
120  tx_buffer="LED1 ON               ";
   break;

   case(6):
```

```
     LED1_OFF();
125  tx_buffer="LED1 OFF                ";
     break;

     case(7):
     LED2_ON();
130  tx_buffer="LED2 ON                 ";
     break;

     case(8):
     LED2_OFF();
135  tx_buffer="LED2 OFF                ";
     break;

   /* POWER */
     case(20):
140  LNA_PWR_ON();
     tx_buffer="LNA POWER ON             ";
     break;

     case(21):
145  LNA_PWR_OFF();
     tx_buffer="LNA POWER OFF             ";
     break;

     case(22):
150  HPA_PWR_ON();
     tx_buffer="HPA POWER ON             ";
     break;

     case(23):
155  HPA_PWR_OFF();
     tx_buffer="HPA POWER OFF             ";
     break;

   /* RADIO FRONT-END */
160
     case(30):
     RX_MODE();
     tx_buffer="RX MODE                 ";
     break;
165
     case(31):
     TX_MODE();
     tx_buffer="TX MODE                 ";
     break;
170
     case(32):
     dac_enable();
     set_dac_output(_2V6);
     tx_buffer="GAIN 2V6                 ";
175  break;

     case(33):
     dac_enable();
     set_dac_output(_2V7);
180  tx_buffer="GAIN 2V7                 ";
     break;

     case(34):
     dac_enable();
185  set_dac_output(_2V8);
     tx_buffer="GAIN 2V8                 ";
     break;

     case(35):
190  dac_enable();
     set_dac_output(_2V9);
     tx_buffer="GAIN 2V9                 ";
     break;

195  case(36):
     dac_enable();
     set_dac_output(_3V0);

     tx_buffer="GAIN 3V0                 ";
200  break;

     case(37):
     dac_disable();
     tx_buffer="GAIN OFF                 ";
205  break;
```

```
/* TRANSCEIVER */

210



/* TEST SETUP */
215
    case(70):
     send_packet(TELEMETRY, buffer, 64);

     tx_buffer="One packet sent                ";
220    break;

    case(71):

    break;
225
    case(72):
    if(TEST72)
    {
     TEST72 = false;
230  tx_buffer="Stoped sending packets..        ";

  HPA_PWR_OFF();
  TX_MODE();
    //LNA_PWR_ON();
235   }

    else
    {
     TEST72 = true;
240   tx_buffer="Sending packets..              ";

  LNA_PWR_OFF();
  TX_MODE();
    HPA_PWR_ON();
245
  _delay_ms(200);

   }

250  break;

    case(73):
     reset = true;
     tx_buffer="System reset..               ";
255  break;

    case(74):
     tx_buffer="Rx mode                      ";

260  break;

   case(75):
    LNA_PWR_OFF();
  TX_MODE();
    HPA_PWR_ON();
265
    delay_ms(200);

  cc1101_tx_mode();
270   tx_buffer="Continious signal             ";

    break;


275  case(76):
     cc1101_reset_transceiver();
  LNA_PWR_OFF();
  HPA_PWR_OFF();
  RX_MODE();
280  cc1101_config();
     cc1101_idle_mode();
     tx_buffer="Transceiver reset            ";

    break;
285
    case(77):
    if(TEST77)
    {
     TEST77 = false;
290  tx_buffer="Stoped sending beacon..        ";
     HPA_PWR_OFF();
```

```
    }

    else
    {
     TEST77 = true;
     tx_buffer="Sending beacon              ";

   LNA_PWR_OFF();
   TX_MODE();
     HPA_PWR_ON();

    }

    break;

    case(78):
     tx_buffer="Beacon sent            ";

    break;


    default:
     status = FAILED;
    break;
   }
  }


  usart_write_table(tx_buffer, 20);


  usart_new_line();

  if(reset)
  {
   system_reset();
  }

  sei();
  return status;
}
```

./firmware/debug.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************************
 *
 * \brief
 *      DAC driver header file.
 *
 *      This file contains the function declarations and definitions for the
 *    debug driver.
 *
 *
 * \par
 *
 *
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:      \n
 *
 *
 *****************************************************************************/


#ifndef _DEBUG_DRIVER_
#define _DEBUG_DRIVER_


int8_t command_call(void);
```

```
39
#endif  /* _DEBUG_DRIVER_ */
```

./firmware/usart.c

```
3
/* This file has been prepared for Doxygen automatic documentation generation.*/
8 /*! \file *********************************************************************
 *
 * \brief
 *      USART driver source file.
 *
13 *      This file contains the USART function for the XMEGA MCU
 *
 *
 * \par
 *
18 *
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
23 *
 * $Revision:
 * $Date:          \n
 *
 *
28 *********************************************************************************/

/******************  INCLUDES  *****************/

33 #include "global_def.h"
#include "usart.h"
#include "mcu.h"
#include "debug.h"

38 /*************************************************/


/******************  DEFINITIONS *****************/
43

/*************************************************/

48


/*******************  MACROS  *******************/
#define Is_usart_tx_buffer_empty()  ((USART_MODULE.STATUS & USART_DREIF_bm) ?
     true:false)
53
/*************************************************/


58
/******************   FUNCTIONS   *****************/


63 /*************************************************/



68
/******************  VARIABLES *****************/

/*************************************************/
```

```
73
78
    /*! \brief Initialize USART module,
     *
     *  This function initializes the hardware on the MCU required to operate the
          USART interface.
     *
83   *  \param
     *
     *  \return Status code
     *  \retval (uint8_t)OK / (uint8_t)FAILED
     */
88  int8_t usart_init(void)
    {
      USART_PORT.DIRSET = PIN7_bm;
      USART_PORT.DIRCLR = PIN6_bm;

93    /* 8bit, 1 stop bit and no parity bit */
      USART_MODULE.CTRLC = USART_CHSIZE_8BIT_gc | USART_PMODE_DISABLED_gc;

      /* set baudrate */
      USART_MODULE.BAUDCTRLA = 103; // 9600 bps at 16 MHz clock
98
      /* enable receive irq */
      USART_MODULE.CTRLA = USART_RXCINTLVL_LO_gc;
        enable_low_level_irq();

103   /* enable RX and TX */
      USART_MODULE.CTRLB = USART_RXEN_bm | USART_TXEN_bm;



108   return OK;
    }



113 /*! \brief Writes a byte to USART,
     *
     *  This function write a single byte onto the USART interface.
     *
     *  \param uint8_t byte
118  *
     *  \return Status code
     *  \retval (uint8_t)OK / (uint8_t)FAILED
     */
    int8_t usart_write_byte(uint8_t byte)
123 {
      while(!Is_usart_tx_buffer_empty());
        USART_MODULE.DATA = byte;

       return OK;
128 }


    /*! \brief Reads a byte from USART,
     *
133  *  This function reads a single byte from the USART interface.
     *
     *  \param none
     *
     *  \return Status code
138  *  \retval (uint8_t)USART buffer
     */
    uint8_t usart_read_byte(void)
    {

143   return  USART_MODULE.DATA;


    }

148
    /*! \brief Writes bytes to USART,
     *
     *  This function write a table onto the USART interface.
     *
153  *  \param uint8_t *usart_buffer,
     *  \param uint8_t bytes
     *
```

```
 *   \return Status code
 *   \retval (uint8_t)OK / (uint8_t)FAILED
 */
int8_t usart_write_table(uint8_t *usart_buffer, uint8_t bytes)
{
  usart_new_line();
  uint8_t i;
  for(i=0; i<bytes; i++)
  {
   usart_write_byte(usart_buffer[i]);
  }

   return OK;
}


ISR(USARTC1_RXC_vect)
{
 volatile uint8_t byte = usart_read_byte();

  if(byte == '#')
   {
    usart_new_line();
    usart_write_byte('#');
    command call();
   }

  else usart_write_byte(byte);
}
```

./firmware/usart.c

```
3
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file *********************************************************************
 *
 * \brief
 *      USART driver source file.
 *
 *      This file contains the USART function for the XMEGA MCU
 *
 *
 * \par
 *
 *
 * \author
 *      Johan L. Tresvig \n
 *      email: j.l.tresvig@fys.uio.no
 *
 * $Revision:
 * $Date:          \n
 *
 *
 *********************************************************************************/

/******************  INCLUDES  *****************/

#include "global_def.h"
#include "usart.h"
#include "mcu.h"
#include "debug.h"

/***********************************************/


/******************  DEFINITIONS *****************/


/***********************************************/

```

```
/********************  MACROS  ********************/
#define Is_usart_tx_buffer_empty()   ((USART_MODULE.STATUS & USART_DREIF_bm) ?
     true:false)
53
/*****************************************************/



58
/******************    FUNCTIONS    ******************/


63 /*****************************************************/



68
/******************  VARIABLES ******************/


/*****************************************************/
73




78
/*! \brief Initialize USART module,
 *
 *  This function initializes the hardware on the MCU required to operate the
     USART interface.
 *
83 *  \param
 *
 *  \return Status code
 *  \retval (uint8_t)OK / (uint8_t)FAILED
 */
88 int8_t usart_init(void)
{
  USART_PORT.DIRSET = PIN7_bm;
  USART_PORT.DIRCLR = PIN6_bm;

93  /* 8bit, 1 stop bit and no parity bit */
  USART_MODULE.CTRLC = USART_CHSIZE_8BIT_gc | USART_PMODE_DISABLED_gc;

  /* set baudrate */
  USART_MODULE.BAUDCTRLA = 103; // 9600 bps at 16 MHz clock
98
  /* enable receive irq */
  USART_MODULE.CTRLA = USART_RXCINTLVL_LO_gc;
    enable_low_level_irq();

103  /* enable RX and TX */
  USART_MODULE.CTRLB = USART_RXEN_bm | USART_TXEN_bm;



108  return OK;
}



113 /*! \brief Writes a byte to USART,
 *
 *  This function write a single byte onto the USART interface.
 *
 *  \param uint8_t byte
118 *
 *  \return Status code
 *  \retval (uint8_t)OK / (uint8_t)FAILED
 */
123 int8_t usart_write_byte(uint8_t byte)
{
  while(!Is_usart_tx_buffer_empty());
    USART_MODULE.DATA = byte;

   return OK;
128 }


/*! \brief Reads a byte from USART,
 *
```

```
133  *   This function reads a single byte from the USART interface.
     *
     *   \param none
     *
     *   \return Status code
138  *   \retval (uint8_t)USART buffer
     */
    uint8_t usart_read_byte(void)
    {

143   return  USART_MODULE.DATA;


    }

148
    /*! \brief Writes bytes to USART,
     *
     *   This function write a table onto the USART interface.
     *
153  *   \param uint8_t *usart_buffer,
     *   \param uint8_t bytes
     *
     *   \return Status code
     *   \retval (uint8_t)OK / (uint8_t)FAILED
158  */
    int8_t usart_write_table(uint8_t *usart_buffer, uint8_t bytes)
    {
      usart_new_line();
      uint8_t i;
163   for(i=0; i<bytes; i++)
      {
       usart_write_byte(usart_buffer[i]);
      }

168    return OK;
    }


    ISR(USARTC1_RXC_vect)
173 {
     volatile uint8_t byte = usart_read_byte();

      if(byte == '#')
       {
178     usart_new_line();
        usart_write_byte('#');
        command_call();
       }

183   else usart_write_byte(byte);
    }
```